Rotation-invariant Face Detection in Grayscale Images

Zhang Wei

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Electronic Engineering

June 2005

Abstract of thesis entitled:
Rotation-invariant Face Detection in Grayscale Images
Submitted by Zhang Wei
for the degree of Master of Philosophy
at The Chinese University of Hong Kong in June 2005

The real-time face detector is an essential part of many applications in, for examples, intelligent human-machine interactive systems, video surveillance, video conferencing, and multimedia indexing and retrieval. This thesis proposes a two-stage, feature-based approach to face detection in grayscale images, which is suitable for real-time implementation.

The proposed detector can detect in-plane rotated human faces in uncontrolled backgrounds. In the first stage, the Canny edges of an image are turned into smooth curves, and then merged to form ellipses. This avoids the necessity of exhaustive searches in the image pyramid that are needed in many present methods. In the second stage, the detector locates a box of facial features in each face candidate ellipse, conditions and then binarizes its contents, and then searches for eye-mouth triangles in the binarized box. A face detection occurs when a positioned eye-mouth triangle matches a reference face model, which also gives the facial feature locations. Experimental results are provided to illustrate the performance of this new detector. Possible improvements on this detector are also discussed.

中文摘要

即時人臉偵測在許多實際應用中都是一個至關重要的部分，例如，智能人際交互系統，視頻監視，視頻會議，以及多媒體檢索。本論文建議了一個兩階段，基於特徵的方法在灰階圖像中進行人臉偵測，此方法適合于即時實現。

此偵測器能夠在不受控制的背景中偵測到多個直立或不直立的人臉。在第一階段，圖像中的 Canny 邊緣被轉化成爲平滑曲綫，然後結合形成多個橢圓。這樣避免了許多現有方法中所必需的層叠式圖像徹底搜尋。在第二階段，該偵測器在每個候選的橢圓中定位一個包含臉部特徵的格子，處理好它的狀態後將其二元化，然後在二元化的格子中搜尋眼睛和嘴所形成的三角。當一個放置好的三角匹配一個參考的人臉模型，一個人臉就被偵測到了，同時可以提供臉部特徵的位置。實驗結果闡明了這個新型偵測器的性能。結論篇提及了此偵測器可能的改進。

Acknowledgement

I would like to thank my supervisor, Professor CHAN Yiu Tong, for guiding me to completions of my research and this thesis with his insights, instructions and helpful suggestions. I also want to express my sincere appreciation to him for proofreading this thesis.

I hope to express my deepest gratitude to my parents for their continuous encouragement and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Face detection is easy for humans but difficult for computers. Machine face detection has received increasing attention lately due to the many potential applications in, for examples, intelligent human-machine interactivity, video surveillance, video conferencing, content-based video coding, and multimedia indexing and retrieval. All of these require automatic and sometimes real-time face detection.

Different applications provide different sources of information for face detection. For example, in conventional fixed-lens surveillance systems, since the background is often unchanged, simply subtracting the reference background produces the region of interest for face detection.

If the background is uncontrolled, face detection in video or image sequences can use the motion information, for example, optical flow [1, 2] and spatiotemporal gradient [3], to help locate the faces.

Face detection in color images has similar advantages. A spectrographic analysis shows that the face skin pixels are usually clustered in a color space. Thus a skin-color segmentation can isolate faces from the background, or, at least, greatly reduce the amount of information which must be processed during the successive stages [4, 5, 6, 7]. The color information of hair [8] and facial features [2, 9] can also be used to detect faces in

color images.

If both the motion and color information are available, they can be combined to achieve a better performance of detection [2, 10]. In the application level, any control on the background or the illumination condition, or any restriction on the number, pose or facial expression of the face can further simplify the face detection problem.

In some applications, the motion and color information are not available. Most of the surveillance cameras nowadays installed in shops, banks and airports are still grayscale cameras due to the lower cost. An extensible image retrieval system must also include grayscale images. To avoid any *a priori* constraint or extra demand on information, this research aims at detecting human faces at the elementary level of source - grayscale images.

1.1 Previous work

Early efforts on face detection [11, 12] assumed a typical passport photograph scenario - plain background and frontal face. The growth of research interest remains stagnant until the 1990's [13], when practical face recognition and video coding systems started to become a reality.

Due to the demand in the rapidly growing computer vision and multimedia technologies, efficient face detection in still images and videos becomes important. Over the past fifteen years, a wide variety of techniques for face detection have been proposed [14, 15], ranging from low-level information (edges, color, motion, etc.) analysis to composite approaches using pattern recognition methods. Among them, the techniques working on still grayscale images are often outperformed by those that detect faces in color images or video or image sequences. The latter have higher accuracy and speed, and can detect rotated and multiple-view faces. This is because color and motion provide

more information for detection.

According to the different approaches to utilizing face knowledge, the face detection techniques working on grayscale images could be broadly classified into two categories. In the first category, face detection is considered as a general recognition problem and face knowledge is incorporated implicitly into the system through mapping and training schemes. Since learning algorithms are often required in these approaches, we name them as *Learning-based* approaches. In the other category, learning is not required and the face knowledge is used explicitly where the low level features are derived prior to knowledge-based analysis. Since the features are the main ingredients in these approaches, we name them as *Feature-based* approaches.

1.1.1 Learning-based approaches

Most learning-based approaches rely on techniques from statistical analysis and machine learning to find the relevant characteristics of face and nonface. Some representative techniques include Distribution-based Models [16], Mixture of Linear Subspaces [17], Sparse Network of Winnows (SNoW) [18], Naive Bayes Classifier [19, 20], Neural Networks [21, 22, 23], Support Vector Machines (SVM) [24, 25] and Boosting Algorithms [25, 26, 27, 28, 29].

Despite a variety of techniques, the learning-based approaches share two common properties:

1. To account for the positional and scale variations of faces in the images, the detectors always search the image pyramid exhaustively to produce sub-windows as face candidates for classification.

2. Models (e.g. face models and facial features models) are learned from a set of training images which should capture the representative variability of facial appearances. These learned

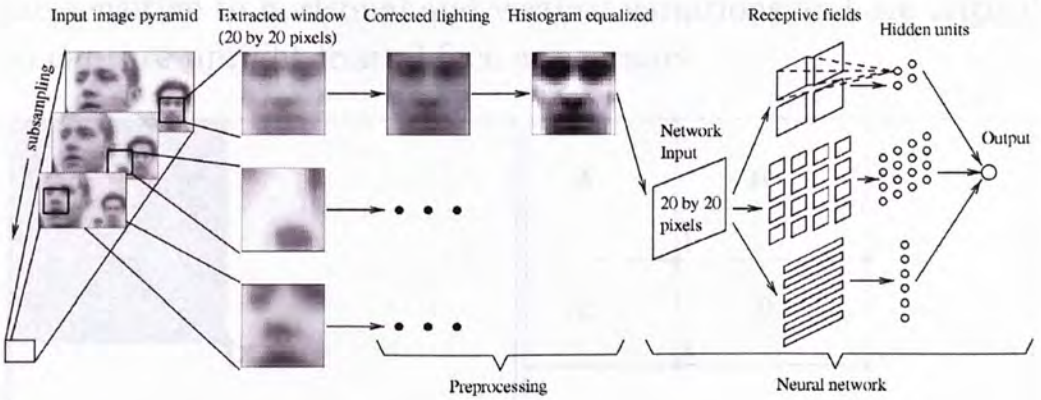


Figure 1.1: A system diagram which contains an image pyramid

models are then used for detection. Exhaustive search in the image pyramid produces a very large number of sub-windows. As in the system of [21], for each step in the pyramid, the input image is scaled down by a factor of 1.2 and scanned by a 20×20 pixel window at every pixel position (see Figure 1.1 [21]). On a test set that contains 130 images and 507 frontal faces, this system needs to examine over 83 million 20×20 pixel windows.

To speed up the classification process on all the sub-windows, [20] used a coarse-to-fine strategy where the likelihood ratio of each sub-window containing the face was first evaluated at a low resolution and only those promising candidates are evaluated at higher resolutions.

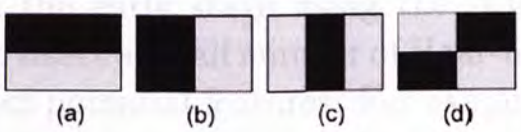


Figure 1.2: Four basic Haar-like features.

Haar-like features are widely used in the learning-based approaches. The four basic Haar-like features, as shown in Figure 1.2 [26], are computed by the mean value difference between pixels in black rectangles and pixels in the gray rectangles. They

are sensitive to horizonal and vertical variations and are critical to capture upright frontal face appearance.

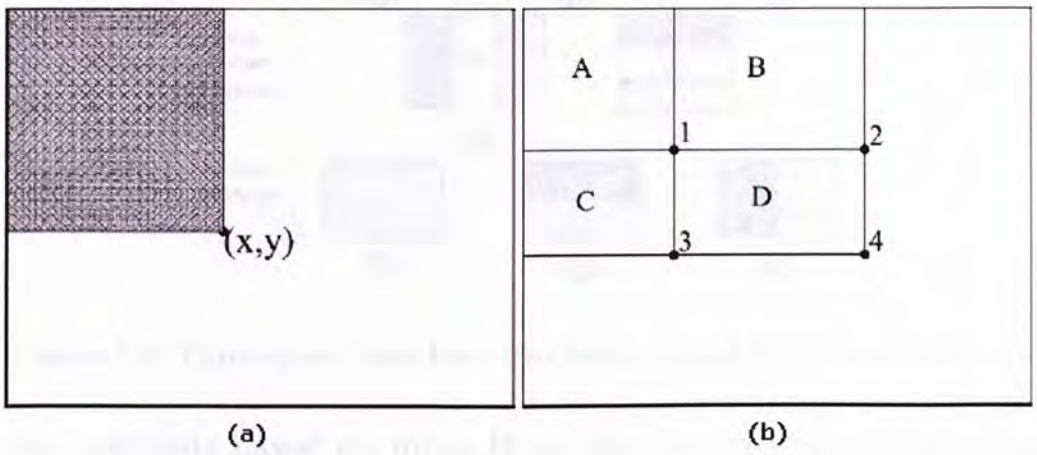


Figure 1.3: The integral image.

The "integral image", first introduced in [26], is an intermediate representation of image which allows the Haar-like features to be computed very quickly. As shown in Figure 1.3(a), the value of the integral image ii(x, y) at point (x, y) is the sum of all the pixels above and to the left. Then the sum of the pixels in rectangle D in Figure 1.3(b) can be computed as $ii(4)+ii(1)-ii(2)-ii(3)$, which means any rectangular sum can be computed in four references in the integral image.

Based on the integral image, a simple and efficient classifier can be built at the early stage using the AdaBoost learning algorithm [30] to select a small number of Haar-like features from a very large set of potential features. For example, [27] selected 2 from a total of 160,000 features where the sub-window size is 24×24. With very few operations (about 60 microprocessor instructions) on the integral image, this two-feature classifier can significantly reduce the number of sub-windows that need further processing while detecting almost all positive instances (detect 100% of the faces with a false alarm rate of 50%).

To detect the rotational and multi-view faces, pose estima-

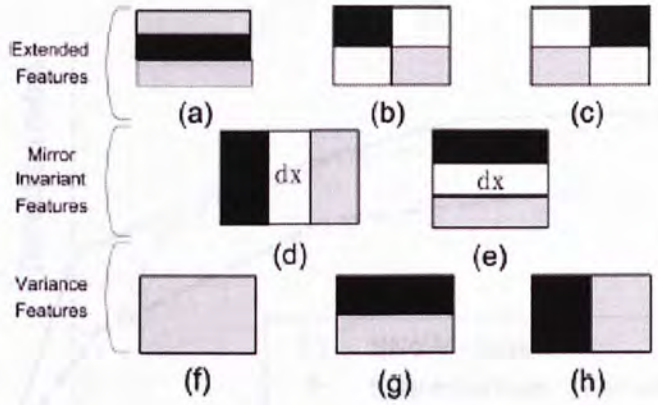


Figure 1.4: Three sets of new Haar-like features used for pose estimation.

tion methods based on more Haar-like features (see Figure 1.4 [25]) are used in [25] and [29] recently. With the help of the integral image, these pose estimation methods are much less computational expensive than some earlier attempts where a pose estimator is separately trained to rotate the sub-window to an upright position [21] or the face/nonface classification is directly processed on each rotated version of the sub-windows [31].

Table 1.1: Some common test sets for face detection.

Data Set	Description
MIT Database	Faces of 16 people, 27 of each person under various illumination conditions, scales and head orientation.
MIT+CMU Test Set	130 grayscale images with a total of 507 frontal faces.
CMU Profile Face Test Set	208 grayscale images with faces in profile views.

Performance evaluation on learning-based approaches is feasible because most of these approaches were tested on some common test sets or part of them (see Table 1.1 [15]). Some representative performance evaluations can be found in [14, 15, 25, 29].

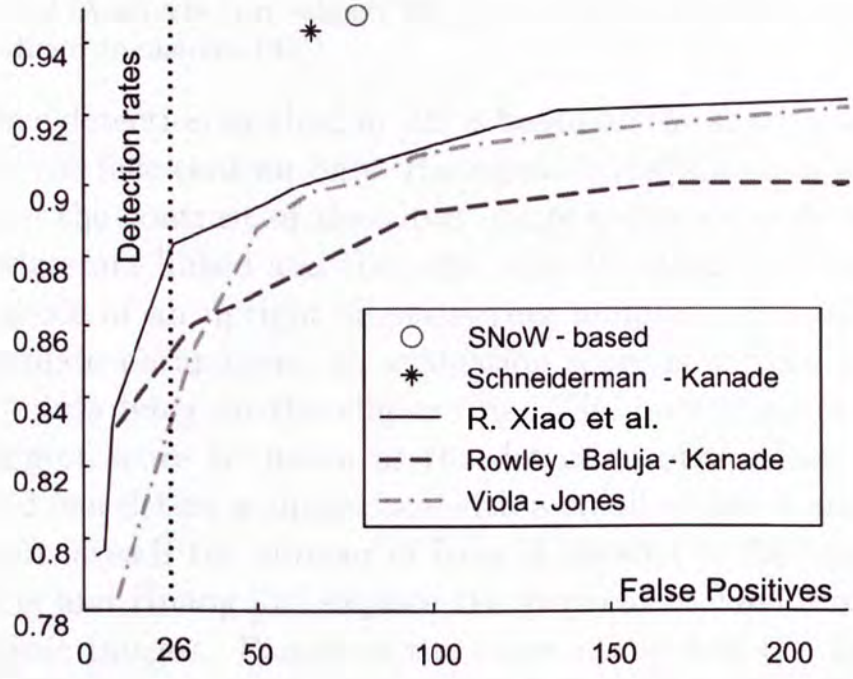


Figure 1.5: The ROC curves of some learning-based approaches.

Recently, R. Xiao et al. [25] compared their detector with some other learning-based ones by the Receiver Operating Characteristic (ROC) curves (see Figure 1.5), where Xiao's detector can achieve a detection rate of over 88% with 26 false alarms.

1.1.2 Feature-based approaches

Feature-based approaches exploit the apparent properties of the face such as shape and face geometry at different system levels. Two most recent surveys on face detection ([14] in 2001 and [15] in 2002) described many different techniques involved in feature-based approaches. We summarize some of them below.

Low-level Analysis (on edges [32], gray information [33, 34], and generalized measures [35])

The face detection method in [32] is based on the shape information of the face contour only. Histogram equalization is used to improve the contrast of the input image before edge detection. The edges are linked and then the edge directions are matched with those of an upright elliptical ring template. For each set of template parameters, an evaluation score is defined for the edge points lying on the ellipse ring. The parameter set with the largest score is chosen as the detection of the face. This method can detect a single face with a small rotation angle (or multiple faces if the number of faces is known) in the image.

Yang and Huang [33] explore the grayscale behavior of faces in mosaic images. Based on the observation that the face region will become uniform when the resolution of a face image is reduced gradually, Yang proposed a hierarchical face detection framework. Starting at low resolution images, face candidates are established by a set of rules that search for uniform regions. The face candidates are then verified by the existence of prominent facial features at higher resolutions. The technique of Yang and Huang was incorporated into a system for rotation invariant face detection by Lv et al. [34] where the local orientation of gradients was analyzed to find the face principal axis.

Reisfeld and Yeshurun [35] introduced a generalized symmetry operator that is based on edge pixel operation. The symmetry measure assigns a symmetry magnitude at every pixel location in an image based on the contributions of surrounding pixels. Since facial features are symmetrical in nature, the symmetry magnitudes at the corresponding locations are high. Hence a geometric analysis on the local maxima gives the detection of a single face in the image.

Active Shape Models (active contour or snakes [36], and deformable templates [37])

Gunn and Nixon [36] introduced a parameterized snake model for face and head boundary extraction. The model consists of dual integrated snakes, one expanding from within the face and the other shrinking from outside the head boundary. The evolution of a snake is achieved by minimizing an energy function. Gunn and Nixon make the energy function sensitive to the image gradient so that the snake is convergent toward edge locations. This method can detect a single face in the image but the dual snakes need to be initialized in the proximity of the face contour.

Yuille et al. [37] used deformable templates to model facial features that fit an a priori elastic model to facial features (e.g. eyes). In this approach, facial features are described by parameterized templates. An energy function is defined to link edges, peaks, and valleys in the input image to corresponding parameters in the templates. The best fit of the elastic model is found by minimizing an energy function of the parameters and this gives the detection of a single face in the input image.

Feature Searching [38]

The facial feature extraction algorithm by De Silva et al. [38] starts by hypothesizing the top of a head and then a searching algorithm scans downward to find an eye-plane which appears to have a sudden increase in edge densities. A flexible facial template covering features such as the eyes and the mouth is initialized on the input image based on the length between the top and the eye-plane. The flexible template is then adjusted to the final feature positions according to a fine-tuning algorithm that employs an edge-based cost function. This method can detect quasi-frontal head and shoulder faces on a plain background.

Constellation Analysis [39, 40]

Yow and Cipolla's [39] algorithm detects feature points from the image using spatial filters and groups them into face candidates using geometric and gray level constraints. A probabilistic framework is then used to reinforce probabilities and to evaluate the likelihood of the candidate as a face. The algorithm is able to cope with small variations in scale, orientation, and viewpoint. A 92% detection rate in 100 lab scene images was reported.

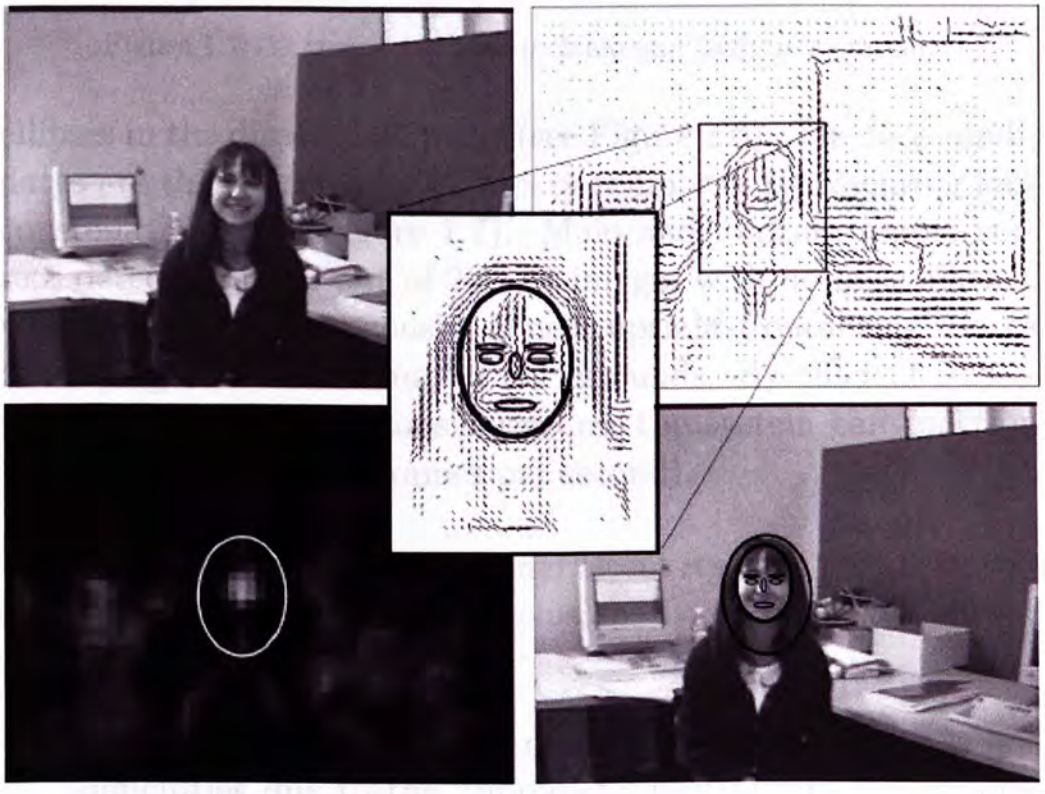


Figure 1.6: Intermediate steps of Maio and Maltoni's system.

In the system of Maio and Maltoni [40], the input images are converted to a directional image using a gradient-type operator over local windows (7×7 pixels) and a generalized Hough transform is used to generate face candidates by searching for

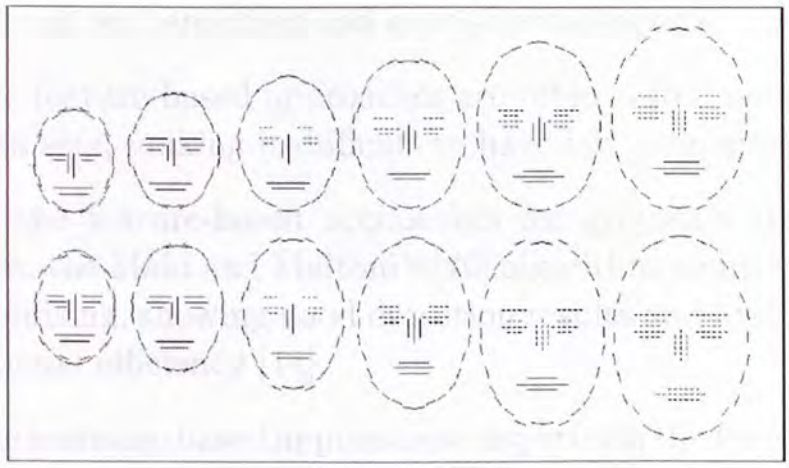


Figure 1.7: 12 templates used in Maio and Maltoni's system.

ellipses in the directional image (see Figure 1.6). The face candidates are then verified by 12 binary templates representing face constellations (see Figure 1.7). Maio and Maltoni report correct detection in 69 out of 70 test images with no false alarms, where the test images consist of near upright-frontal single faces of varying sizes with complex backgrounds. By efficient implementation and design considerations, the system can function in real-time (about 13 frames per second).

As there is no significant additional recent research that is feature-based, we summarize on the conclusions drawn in the two survey papers.

- Face detection by explicit modelling of facial features has difficulties due to the unpredictability of face appearance and environmental conditions (in contrast to the learning-based approaches which formulate the problem as one of learning to recognize a face pattern from examples, so that the specific application of face knowledge is avoided).
- Feature-based approaches are designed mainly for face localization (single face detection), and most are still limited

to head and shoulder and quasi-frontal faces.

- The feature-based approaches are often tested on different data sets, making it difficult to have fair comparisons.
- Of the feature-based approaches for grayscale static images, the Maio and Maltoni's [40] algorithm seems the most promising, showing good detection results and high computational efficiency [14].
- The learning-based approaches outperform the feature-based ones in terms of robustness. (The most recently proposed learning-based approaches [25, 28, 29] are also efficient.)
- *"However, since an exhaustive window scanning in the image pyramid is not always preferable, feature-based methods can provide visual cues to focus attention."* quoting [14].

1.2 Thesis objective

In the literature, three recently developed learning-based approaches [25, 28, 29] based on Haar-like features and Boosting Algorithms reported real-time detection of multi-view faces in single grayscale image with promising accuracy. However, there are two factors that may cause deficiencies in these learning-based approaches.

1. The detection begins with an exhaustive search in the image pyramid and relies on the content in the rectangular subwindows. As a result, some naturally occurring "nonface" patterns (see Figure 1.8 [16]) can be classified as faces.
2. The long training time is usually ignored, but it may be important for real-time applications that require online training on different data sets [15].

Figure 1.8: An example of a naturally occurring "nonface" pattern that resembles a face. Left: Viewed in isolation. Right: Viewed in the context of its environment.

To compensate for the first deficiency, we utilize the shape information of face contour. On the other hand, we only use feature-based approaches and avoid any training process.

The previous feature-based approaches using shape information (e.g. [32] and the Maio and Maltoni's [40] algorithm) have problems in detecting multiple rotated faces, while the only feature-based approach in the literature that can detect multiple rotated human faces [34] (the system should be able to detect multiple faces though it is not clearly reported) ignored the shape information of face contour, so that it still has the first deficiency.

In this thesis, we propose a feature-based face detector which can detect multiple rotated frontal faces in grayscale images. It is also computationally efficient and suitable for real-time implementation. We will test the proposed detector on the images from the common test sets in Table 1.1 for easy evaluation.

1.3 The proposed detector

A new face detector is proposed which locates frontal views of human faces with arbitrary in-plane rotations in complex scenes. Figure 1.9 shows the overview of the proposed detector. In the first stage, the detector locates the face candidates by an edge

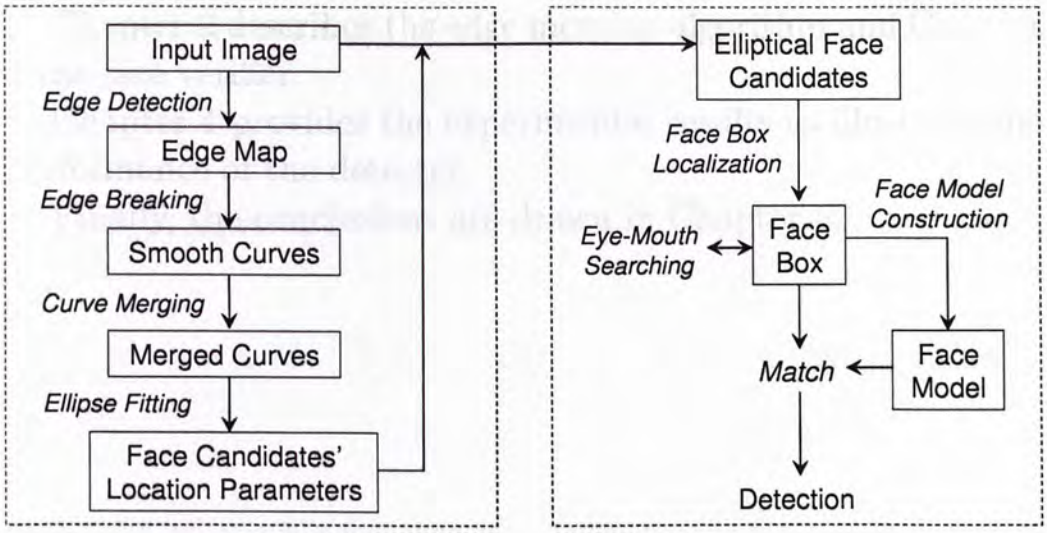


Figure 1.9: Overview of the proposed detector

merging and ellipse fitting algorithm. The algorithm utilizes the shape information of face contour by conditioning Canny edges of the input image, hence avoiding the exhaustive searches in the image pyramid. In the second stage, the detector verifies each ellipse-localized face candidate by comparing the face box that contains only the facial features to a binary deformable face model. To cover the case that the whole image is one human face without silhouette, the whole input image is always first tested by the second stage. The facial features are located once the face is detected, thus facilitating further analysis such as face recognition and facial expression recognition.

Since both the edge merging and the face verification process can be done in parallel, and due to simplicity at the algorithm level, this detector has a high potential for real-time implementation.

1.4 Thesis outline

The rest of the thesis is organized as follows:

Chapter 2 describes the edge merging algorithm and Chapter 3 the face verifier.

Chapter 4 provides the experimental results to illustrate the performance of the detector.

Finally, the conclusions are drawn in Chapter 5.

□ **End of chapter.**

Chapter 2

The Edge Merging Algorithm

A novel edge merging algorithm is proposed as the first stage of the detector to locate face candidates by conditioning the Canny edge map of the input image and utilizing the shape information of face contour. The algorithm consists of four steps:

1. Edge detection - detects the Canny edges of the input image to get the edge map.
2. Edge breaking - breaks the edges by eliminating the crosses and the corners in the edge map to get the smooth curves.
3. Curve merging - merges the smooth curves to form the ellipse-like curves.
4. Ellipse fitting - fits the merged curves with ellipses to get the location parameters of face candidates.

2.1 Edge detection

For an image with complex and uncontrolled background, applying global contrast improvements (such as lighting correction and histogram equalization) on the whole image does not necessarily help detect the edges that belong to the face contours.

Here, we apply Canny edge detection directly to the original input image because it detects weak edges that are connected to the strong ones, hence is less sensitive to noise.

The Canny edge detection method [41] finds edges by looking for local maxima of the gradient of the input image. The gradient is calculated using the derivatives (along x- and y-axis) of a 2D Gaussian filter. The filter size is selected to omit responses smaller than 10^{-4}. The standard deviation of the filter is 1 so that the filter size is 9 by 9.

The method uses two thresholds to detect strong and weak edges. The high threshold TH_{high} is selected so that only 30% of the pixels in the image have higher gradient magnitudes than the value of TH_{high}. The low threshold TH_{low} is $0.4\times\mathrm{TH}_{high}$. The local maxima of the gradient magnitudes larger than TH_{high} are the strong edges and are set to be 1 in the output edge map. Those between TH_{low} and TH_{high} are also included in the output if they are 8-connected to the strong edges.

Figure 2.1 shows the edge maps of the image "Lena", detected by the Sobel method, the Laplacian of Gaussian (LoG) method, and the Canny method respectively. We can observe that, by linking the weak edges to the strong ones, the Canny edge map is relatively complete and continuous, thus providing sufficient information for locating the face (or head) contours. This is the reason for choosing the Canny edge detector in the proposed face detector. In most cases, the setting of providing 30% strong edges works well.

Because the Canny edges are detected by only considering the intensity differences, in many cases, the edges belonging to a face contour are connected to the edges of another face contour, some facial features, or the cluttered background. We need to break these false connections to provide the isolated face contour curves.

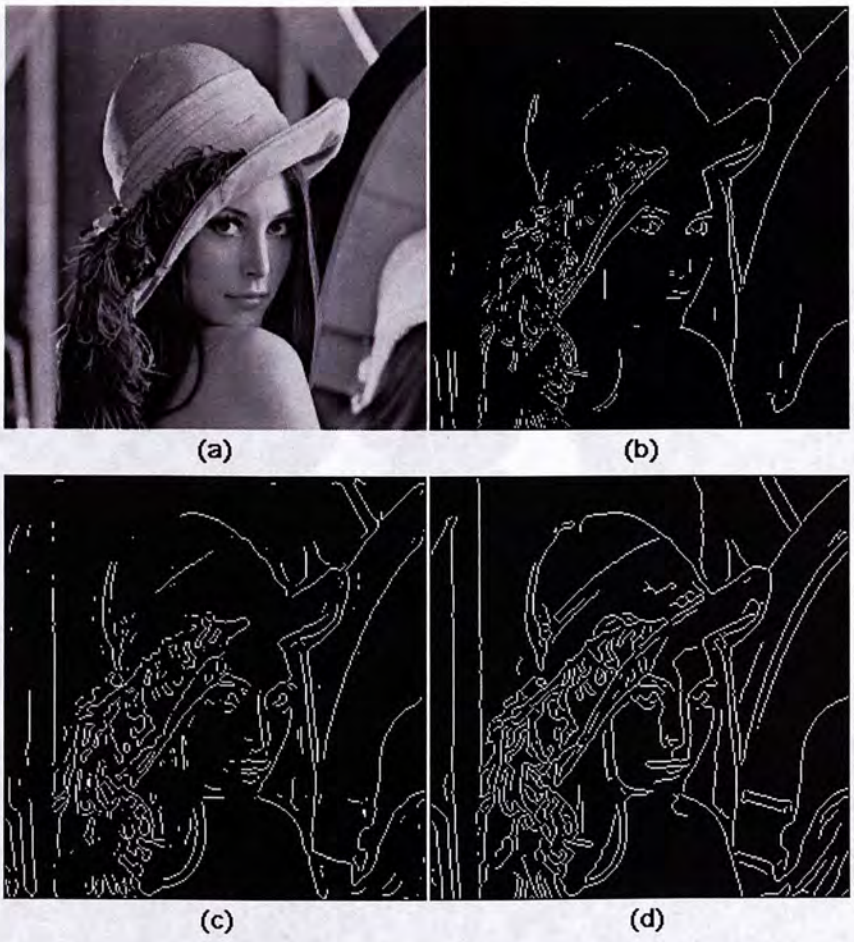


Figure 2.1: Compare the Canny method with other edge detection methods. (a) the original image (b) Sobel edges (c) LoG edges (d) Canny edges

2.2 Edge breaking

Human face and head contours are not self-intersecting but smooth. Based on this observation, we break the false connections by eliminating the crosses and the corners in the edge map.

Figure 2.2 shows the image "cast1" in the CMU Rotated Test Set and its Canny edge map. Consider the two edges in bold blue, where the left one contains a cross (in the red rectangle) and three corners (in the red circles) while the right one contains

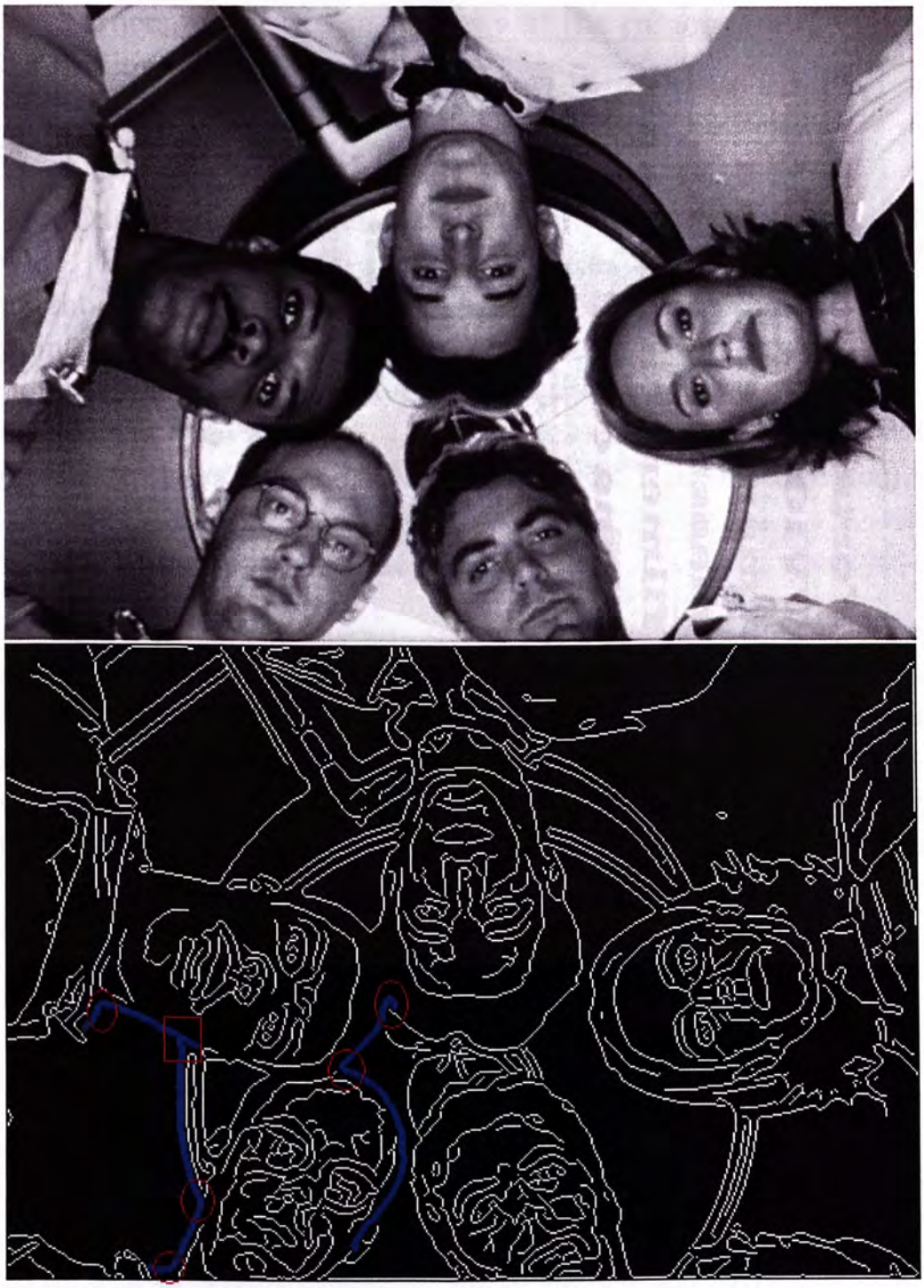


Figure 2.2: The image "cast1" and its Canny edge map

two corners. Clearly, the crosses and the corners are the false connections.

2.2.1 Cross detection

The Canny edges are thinned edges so that they are one pixel wide. For each pixel in a non-intersecting edge, its number of neighbors (in 8-connectivity) must be either one (for the end-point of the edge) or two (for the pixels that are not the end-point). Therefore, the crosses can be easily detected by searching for the edge pixels that have more than two neighbors. We eliminate the crosses by setting them to be 0 in the edge map.

2.2.2 Corner detection

After breaking the crosses, the remaining curves in the edge map are not furcate and can be represented by the chain codes [42]. Associated with eight possible directions, the chain codes are defined as shown in Figure 2.3(a), where x is the current curve pixel position. Each code can be considered as the angular direction that we must move to go from one curve pixel to the next, in multiples of $\pi/4$. Starting from the pixel in dark, the chain code representation of the curve in Figure 2.3(b) is {0, 1, 1, 2, 2, 3, 4, 4, 5, 6, 7}.

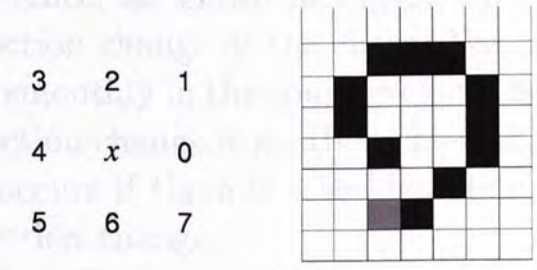


Figure 2.3: (a) Definition of the chain codes (b) A curve that can be represented as chain codes

Figure 2.4: Example of corner detection

The curve changes its direction when there is a change between consecutive chain codes, so the corners can be detected by analyzing the chain codes. Consider the blue curve in Figure 2.4 which has two corners. We represent this curve in chain codes (starting from the circled pixel), as shown in Figure 2.5(a). Figure 2.5(b) shows the difference of the consecutive chain codes (chain code difference), which indicates the direction change of the curve at each pixel. Therefore, the cumulative sum of the chain code difference, as shown in Figure 2.5(c), indicates the cumulative direction change of the curve. We can observe that the curve turns smoothly in the counter-clockwise direction if its cumulative direction change is gently increasing, and vice versa. Also, a corner occurs if there is a sudden climb or drop in the cumulative direction change.

To remove the effects caused by the quantization noise, we average the cumulative sum of the chain code difference on every 5 pixels to get the smoothed cumulative direction change, as shown in 2.5(d). The sudden climbs or drops in the smoothed

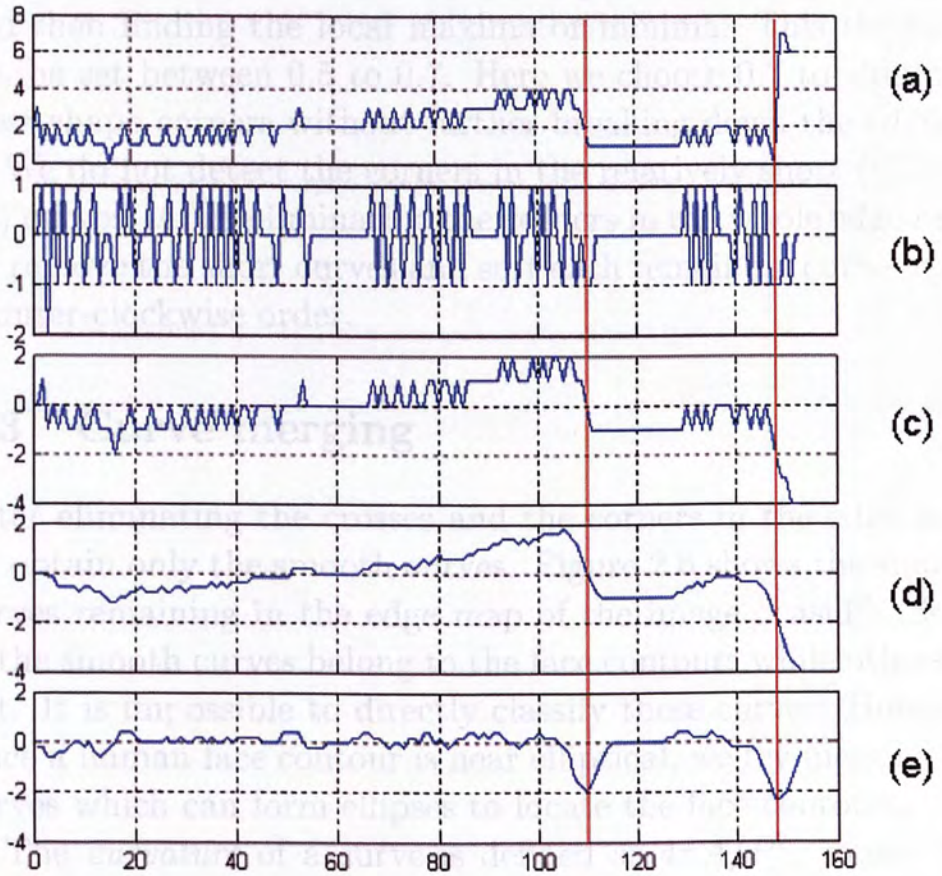


Figure 2.5: Analyze the chain codes, from top to down: (a) chain codes, (b) chain code difference, (c) cumulative sum of chain code difference, (d) averaging (c) on every 5 pixels, (e) difference of (d) at every 5 pixels.

cumulative direction change correspond to the peaks or the valleys in Figure 2.5(e), which is the difference of (d) at every 5 pixels. The positions of the two local minima in Figure 2.5(e) (as indicated by the red vertical line) correspond to the two corners in the curve (as indicated by the red crosses in Figure 2.4). Therefore the corners are detected by thresholding (at ± 0.7) and then finding the local maxima or minima. This threshold can be set between 0.5 to 0.7. Here we choose 0.7 to eliminate most shape corners without further breaking down the edges.

We do not detect the corners in the relatively short ($\leq$5 pixels) curves. After eliminating the corners in the whole edge map, we remove the short curves and sort each remaining curve in the counter-clockwise order.

2.3 Curve merging

After eliminating the crosses and the corners in the edge map, we obtain only the smooth curves. Figure 2.6 shows the smooth curves remaining in the edge map of the image "cast1". Some of the smooth curves belong to the face contours while others do not. It is impossible to directly classify these curves. However, since a human face contour is near elliptical, we try merging the curves which can form ellipses to locate the face contours.

The *curvature* of a curve is defined as $4\pi A/P^2$, where A is the area bounded by the curve and the line joining its endpoints, and P is the perimeter of this area. The value of curvature varies from 0 (straight lines) to 1 (circles).

The curve merging process begins with the long curves whose curvature exceed a predefined threshold (0.05), and continues in the following way:

- Define the search region for the current curve.

Figure 2.6: The smooth curves remaining in the edge map of "cast1"

- For each candidate curve in the search region, a cost is assigned for merging it with the current curve.
- Merge with the candidate curve that has the lowest cost by directly linking the corresponding endpoints.
- Define the new search region and start the next iteration.
- If there is no candidate curve in the search region that can increase the curvature, or the lowest cost exceeds a predefined threshold, the merging process terminates.

The candidate curves with small curvatures (< 0.05) are treated as "straight lines", and they could be merged with the current curve in either the counter-clockwise or clockwise direction. Since the curve merging process will not be applied on the "straight lines", the definition of "straight line" can be relaxed

by increasing the threshold up to 0.15 to merge more curves. In our experiments, we choose 0.05.

Curve merging fails when the curvature of the merged curve is too small (i.e. < 0.8). A merged curve is also discarded if the length of the linkage exceeds 50% of the curve length.

2.3.1 The search region

In each iteration of the curve merging process, a *search region*, as indicated by the white regions in Figure 2.7, is determined according to the length L, the curvature c, and the directions at both endpoints of the current curve.

As shown in Figure 2.7, we determine the search region in the following steps:

- In both cases, the direction at an endpoint is approximated as the direction of the line joining the endpoint (point 1 or 2) and the m-th pixel away from the endpoint (point 3 or 4), where m is the integer of rounding $L/(2+2c)$ towards infinity.
- In both cases, point 10 is the intersection of the line joining points 1 and 3, and the line joining points 2 and 4.
- If point 10 is on the same side of the line joining the endpoints as the whole curve, it falls into the 1st case as shown in Figure 2.7(a).
  - Define a search range, $r = L/(1+c)$, so that point 7 is obtained by extending from one of the endpoints (point 1), in the direction of the endpoint, to the distance of r. Similarly, point 8 is obtained by extending from the other endpoint (point 2).
  - Point 5 is the midpoint of the line joining two endpoints, and point 6 is the midpoint of the current curve.

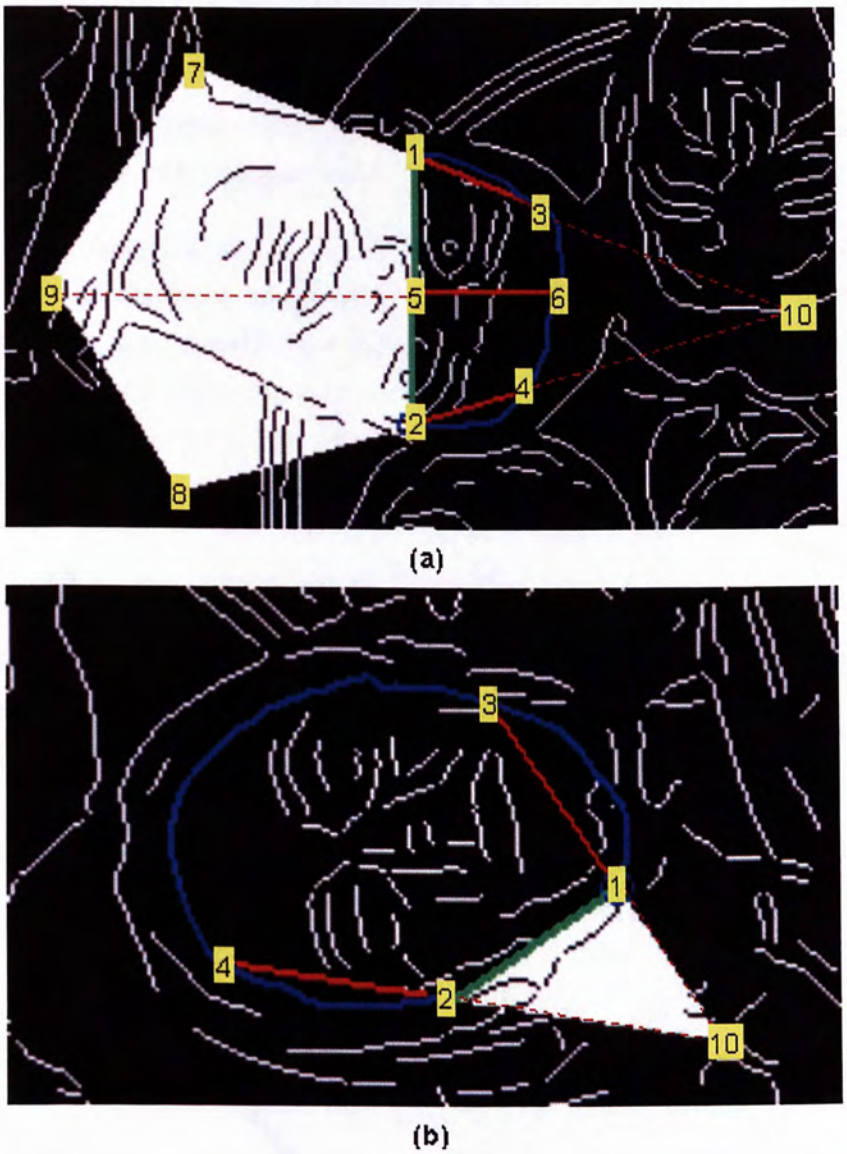


Figure 2.7: The search region - two cases in (a) and (b)

- Point 9 is obtained by extending from point 5, in the direction of the line joining point 5 and 6, to the distance of $1.5r$.
- The search region in this case is a pentagon with points 1, 7, 9, 8 and 2 as its vertices.

- If it is the 2nd case, the search region, as shown in Figure 2.7(b), is a triangle with points 1, 2 and 10 as its vertices.

A candidate curve is considered for merging with the current curve if it is shorter than the current curve and more than half of its pixels are inside the search region. As the curvature of a curve increases, the size of the search region decreases. Thus, as the merging process goes on, the number of curves to be analyzed decreases rapidly.

2.3.2 The merging cost function

To merge with the *best* curve in each iteration, a cost function is defined so that each candidate curve is assigned with a cost and the *best* curve is the one with the lowest cost.

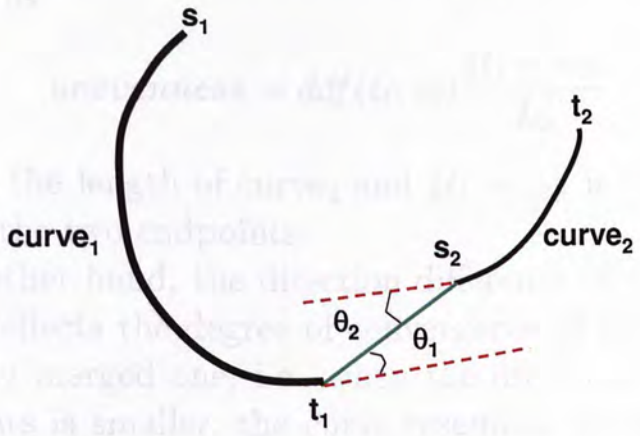


Figure 2.8: An example of merging, curve_1 is the current curve and curve_2 is a candidate curve

Since the curves are all sorted in the counter-clockwise order, we can determine where to merge two curves by comparing two distances. As shown in Figure 2.8, curve_1 is the current curve and curve_2 is a candidate curve. s_1 and s_2 are the starts of the two curves while t_1 and t_2 are the ends. As the distance between t_1 and s_2 is shorter than the distance between s_1 and t_2, we only try merging the two curves at t_1 and s_2.

In Figure 2.8, the red dash lines indicate the endpoint directions of t_1 and s_2 respectively. The green solid line is the line joining t_1 and s_2. The angles between the red dash lines and the green solid line are θ_1 and θ_2 respectively. We define the direction difference, *ddf*, of t_1 and s_2 as the larger of θ_1 and θ_2 (in radian), that is,

$$ddf(t_1, s_2) = max(\theta_1, \theta_2)$$

The direction difference of t_1 and s_2 reflects the *unevenness* of merging at these two endpoints, i.e. the merging at t_1 and s_2 is uneven when $ddf(t_1, s_2)$ is large and vice versa. As we do not wish to merge two endpoints if their distance is long or the length of the candidate curve (curve_2) is short, we define *unevenness* as

$$unevenness = ddf(t_1, s_2)\frac{\|t_1 - s_2\|}{L_2}$$

where L_2 is the length of curve_2 and $\|t_1 - s_2\|$ is the Euclidean distance of the two endpoints.

On the other hand, the direction difference of the endpoints of a curve reflects the degree of convergence of this curve from a completely merged one, i.e. when the direction difference of the endpoints is smaller, the curve resembles more a complete curve. As shown in Figure 2.9, curve_3 is the intermediate merged curve obtained by merging curve_1 and curve_2 in Figure 2.8. It starts from the start of curve_1 (s_1) and terminates at the end

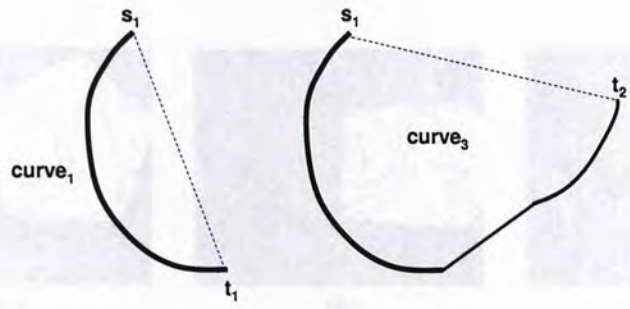


Figure 2.9: Before and after merging, curve_1 is the original curve and curve_3 is the intermediate merged curve by merging curve_1 and curve_2 in Figure 2.8

of curve_2 (t_2). We measure the direction differences of the endpoints of curve_1, $ddf(t_1, s_1)$, and that of curve_3, $ddf(t_2, s_1)$. The *divergency* of this merging can be defined as

$$divergency = \frac{ddf(t_2, s_1)}{ddf(t_1, s_1)}$$

The total merging cost is defined based on the *unevenness* and the *divergency*. We observe from the experiment that *unevenness* is more expensive than *divergency* when the curvature of the current curve is small (at the earlier stages of merging), and vice versa. Therefore, we define the total cost by weighting the two terms and then averaging:

$$cost = \frac{(2 - c_1) \times unevenness + c_1 \times divergency}{2}$$

where c_1 is the curvature of the current curve (curve_1).

In each iteration of curve merging, the candidate curve in the search region that has the lowest cost is merged with the current curve by directly linking the corresponding endpoints, such as t_1 and s_2 in Figure 2.8. The curve merging process terminates if the lowest cost exceeds 2. Figure 2.10 shows a sample curve merging process on the smooth curves of the image "cast1".

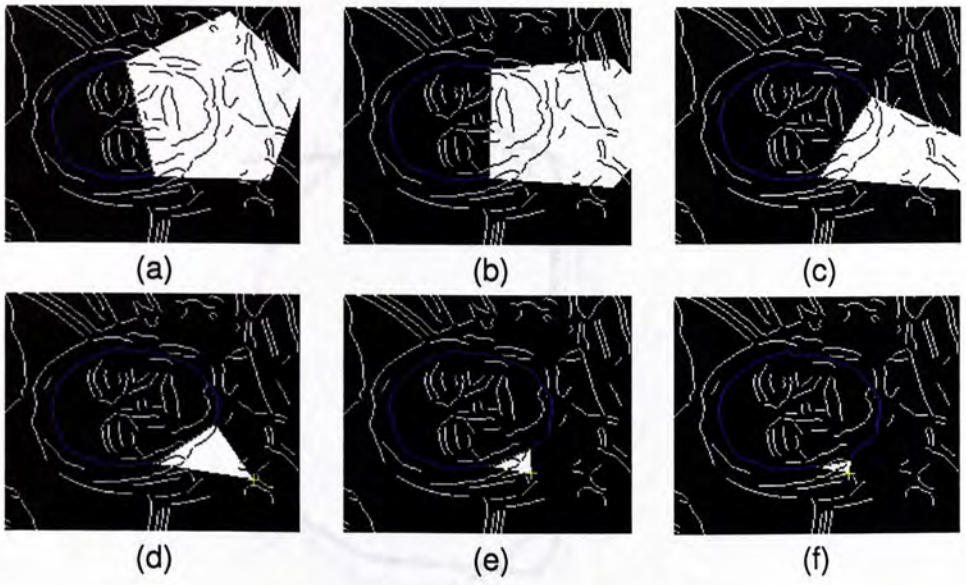


Figure 2.10: A sample curve merging process

2.4 Ellipse fitting

The merged curves are fitted by ellipses for two reasons:

1. Faces are generally elliptical. Thus we fit the merged curves by ellipses and only those with small fitting errors are taken to be the face candidates. Figure 2.11 shows a merged curve that has a large fitting error and its fitted ellipse.

2. Rather than storing all the coordinates of the pixels of the merged curve, ellipse fitting can greatly reduce the data storage, since only 5 parameters (2 for ellipse center, 1 for major axis, 1 for minor axis and 1 for rotation angle) are needed to determine a face candidate.

We use the algebraic ellipse fitting method [43] to fit the merged curves with ellipses in the following way.

- Represent an ellipse in the x-y plane as:

$$F(B,u) = Bu = ax^2 + bxy + cy^2 + dx + ey + f$$

Figure 2.11: A merged curve that has a large fitting error and its fitted ellipse

where $B = [x^2\ xy\ y^2\ x\ y\ 1]$, and $u = [a\ b\ c\ d\ e\ f]^T$ are the parameters of the ellipse.

- $F(B_i, u) = d$ is called the "algebraic distance" of a point (x_i, y_i) to the ellipse $F(B, u) = 0$.
- Solve the constrained least square problem to minimize d, i.e. $||Bu|| = min$ subject to $||u|| = 1$.
- Calculate the ellipse parameters from u.

Fitting error is calculated as the average of the smallest Euclidean distance of each point to the fitted ellipse. We normalize the fitting error by $\sqrt{a^2 + b^2}$, where a and b are lengths of the major and minor axis, to allow the longer curves to have larger fitting errors. An ellipse is discarded if its normalized fitting error exceeds a predefined threshold. We found this threshold can be set between 0.05 to 0.08 from our experiments, here we

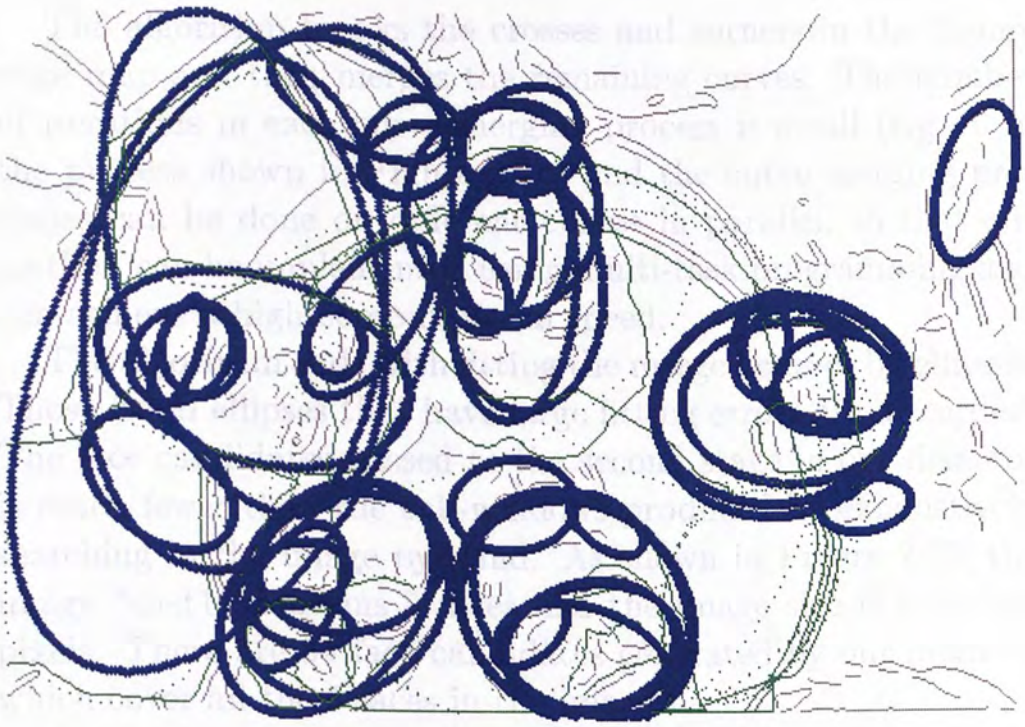


Figure 2.12: The fitted ellipses (in blue) and the rejected curves (in green) of the image "cast1"

choose 0.06. Figure 2.12 shows the fitted ellipses and the rejected curves in the image "cast1".

2.5 Discussion

In this chapter, we have introduced an edge merging algorithm to locate face candidates in the first stage of the proposed face detector.

The algorithm breaks the crosses and corners in the Canny edge map and then merges the remaining curves. The number of iterations in each curve merging process is small (e.g. 6 in the process shown in Figure 2.10) and the curve merging processes can be done on different curves in parallel, so that our method can be implemented using multi-task programming and can achieve a high computational speed.

The algorithm ends with fitting the merged curves by ellipses. Those fitted ellipses that have large fitting errors are discarded. The face candidates passed to the second stage in our detector is much fewer than the sub-windows produced by exhaustively searching in the image pyramid. As shown in Figure 2.12, the image "cast1" contains 5 faces and the image size is 504×350 pixels. There are 54 face candidates generated by our method, which cover all the 5 faces in the image.

In our method, the number of face candidates increases with the number of edges in the image while the number of edges reflects the complexity of the image content. This is more reasonable than exhaustive search in image pyramid, where the number of sub-windows increases with the image size.

Comparing to the ellipse ring template matching on edge directions used in [32] and the Hough transform in the directional image used in [40], our method is different from them, and also outperforms them in the following aspects.

- Both the ellipse template matching and the Hough trans-

form are actually voting processes. In [32], each edge point votes to a template parameter set if it is lying on the ellipse ring. In [40], each vector in the directional image votes to an accumulate array in the parameter space of the Hough transform. After the voting, only the one that has the largest score is chosen as a detected face ([32]) or face candidate ([40]). This is the reason why these two methods are limited to detecting a single face, or multiple faces only if the number of faces is known, in the image.

- Our method detects faces with arbitrary rotations. If the methods in [32] or [40] are extended to detect rotational faces, the parameter space has to be extended to a higher number of dimensions, which will increase the computational cost significantly.
- These two methods may not be able to detect the correct faces when the images have objects (especially when there are many) whose shapes are similar to ellipses, and these cases cannot be avoided in uncontrolled backgrounds.
- The method in [32] is pixel-wise in the edge map. The method in [40] is based on the directional image calculated on every 7×7 pixels in the image, so it is also pixel-wise. Our method is edge-wise and is thus more efficient.

□ **End of chapter.**

Chapter 3

The Face Verifier

In the previous chapter, we have shown that the first stage of the detector utilizes the shape information of face contour and locates elliptical face candidates in the image. However, the face contour alone is not sufficient to model a face. In this chapter, we introduce a face verifier as the second stage of the detector. It verifies a face by checking the existence and proper alignments of facial features, in four steps: It

1. Locates a face box that includes only eyebrows, eyes, nose and mouth, from the elliptical face candidate.
2. Conditions the face box, then binarizes its content.
3. Searches the binarized box for the triangle(s) formed by the eyes (or eyebrows) and the mouth.
4. Matches the triangle against a binary deformable face model, which is adjusted according to the triangle, for detection.

3.1 The face box

An elliptical face candidate should ideally contain only facial features of eyebrows, eyes, nose and mouth. To exclude non-face components such as hair, ears or neck, and to limit the

Figure 3.1: Sample face candidates

verification within a box, we next find a face box from a face candidate ellipse. Some examples of face candidates are in Figure 3.1. (Note that generally the face orientation is close to the ellipse orientation except we have not resolved the up-down ambiguity.) The next section describes the localization of a face box that contains just the facial features.

3.1.1 Face box localization

It is necessary to align the face box properly with respect to the center of the ellipse (face candidate). As the eyes and the mouth are isotropically symmetric, we determine the box width and its horizontal location by projecting the isotropic symmetry measure [35] of a face candidate onto the horizontal axis.

Isotropic symmetry measure

We use the generalized symmetry operator introduced in [35] to measure the isotropic symmetry. The operator assigns a magnitude at every pixel location, based on the contributions of surrounding pixel pairs. We next explain how the magnitude $M(o)$ at a pixel o is assigned, so that a symmetry map, M for the whole image can be obtained.

The symmetry magnitude, $M(o)$ is defined as

$$M(o) = \sum_{(p,q)\in\Gamma(o)} C(p,q)$$

where (p, q) denotes each pixel pair in the *pixel pair set*, $\Gamma(o)$, and $C(p, q)$ is the *contribution* of this pixel pair.

The *pixel pair set*, $\Gamma(o)$ is defined as

$$\Gamma(o) = \{(p, q) \text{ s.t. } \frac{p+q}{2} = o \text{ and } \|o - p\| \leq 3\sigma\}$$

where $\frac{p+q}{2} = o$ means that pixel o is the midpoint of the line joining p and q, $\|o - p\|$ is the Euclidean distance of two pixels, and σ is the standard deviation of the 2D Gaussian function for calculating the gradient of intensity (∇I).

As suggested in [35], $\sigma = (\sigma_x, \sigma_y) = (1, \frac{2}{3})$, and in practice the Gaussian function is sampled and then cut off after 3σ. Therefore, $\Gamma(o)$ is defined in the 7×5 neighborhood of pixel o.

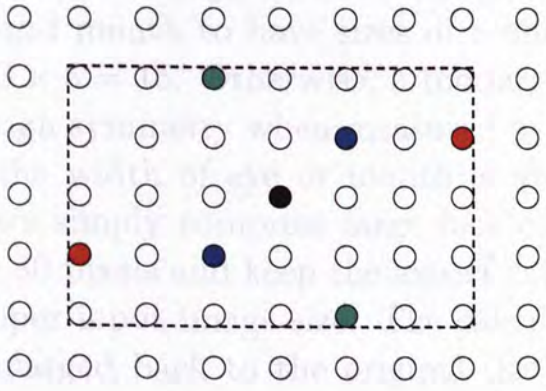


Figure 3.2: Define the pixel pair set $\Gamma(o)$

In Figure 3.2, the black pixel in center denotes the pixel o and the dashed rectangle bounds its 7×5 neighborhood. The pixels in red, blue and green are three pixel pairs in the set $\Gamma(o)$, and there should be $(7 \times 5 - 1)/2 = 17$ such pairs.

The *contribution*, $C(p, q)$ of a pixel pair (p, q) is defined as

$$C(p, q) = D(p, q)P(p, q)r_p r_q$$

where $D(p, q)$ is a distance weight function, $P(p, q)$ is a phase weight function, r_p and r_q denote the logarithm of gradient at

pixel p and q respectively, and the logarithm of gradient is defined as $r_k = log(1 + \|\nabla I_k\|)$, where k is any pixel.

The two weight functions are defined as

$$D(p,q) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{\|p-q\|}{2\sigma}}$$

and

$$P(p,q) = (1 - cos(\theta_p + \theta_q - 2\alpha_{pq}))(1 - cos(\theta_p - \theta_q))$$

where α_{pq} is the counterclockwise angle between the line joining p and q and the horizon, θ_p and θ_q denote the direction of gradient at p and q respectively, and the direction of gradient is defined as $\theta_k = tan^{-1}(\frac{\partial}{\partial y}\nabla I_k / \frac{\partial}{\partial x}\nabla I_k)$, where k is any pixel.

In order to follow the suggested settings (i.e. $\sigma = (1, \frac{2}{3})$), we need the eyes and mouth to have sizes of around or less than $3 \times 7 = 21$ by $3 \times 5 = 15$. Otherwise, a too large eye or mouth will not show high symmetry when measured by this symmetry operator. As the width of eye or mouth is about 1/4 of the ellipse width, we simply compress large face candidates to fix their widths as 80 pixels and keep the aspect ratios unchanged, to ensure a proper input image size. The calculated symmetry map is then enlarged back to the original size by bilinear interpolation. Figure 3.3 shows a face candidate and its isotropic symmetry map M.

It is noted earlier that the orientation of the face is close to that of the ellipse, and that the eyes and the mouth have high horizontal symmetry. Therefore we link the symmetry magnitudes horizontally, using the method in [35], to highlight the responses at the eyes and the mouth, giving a linked symmetry map ML.

This map ML is the multiplication of the left and right accumulated symmetry maps, L and R. We next explain how to calculate L from M, and R can be calculated in the same manner.

(a) (b)

Figure 3.3: (a) An elliptical face candidates and (b) its symmetry map M

There are three parameters involved in calculating L: a global threshold, $T \in [0, 1]$, a local threshold, $t \in [0, 1]$, and a neighbor weight, $w \in [0, 1]$. L is calculated using a dynamic programming procedure:

Let m be the maximum value in M. Going from left to right, for each pixel, $o = (x, y)$, let

$$l(x, y) = max(L(x-1, y-1), L(x-1, y), L(x-1, y+1))$$

where L, the left accumulated symmetry map, is defined as

$$L(o) = \begin{cases} M(o) & \text{if } M(o) < Tm \text{ or } M(o)tm > l(o) \\ M(o) + wl(o) & \text{otherwise} \end{cases}$$

where the suggested parameter set in [35] is used, i.e. $T = 0.01$, $t = 0.7$, and $w = 0.9$.

R is calculated in the same manner and the two accumulated symmetry maps (L and R) are multiplied. In this way the symmetry clusters, which correspond to the eyes, the eyebrows, the mouth, and perhaps the nostrils, are highlighted. Histogram equalization [14] is applied on the linked symmetry map ML to improve the contrast, which results in ML_{eq}.

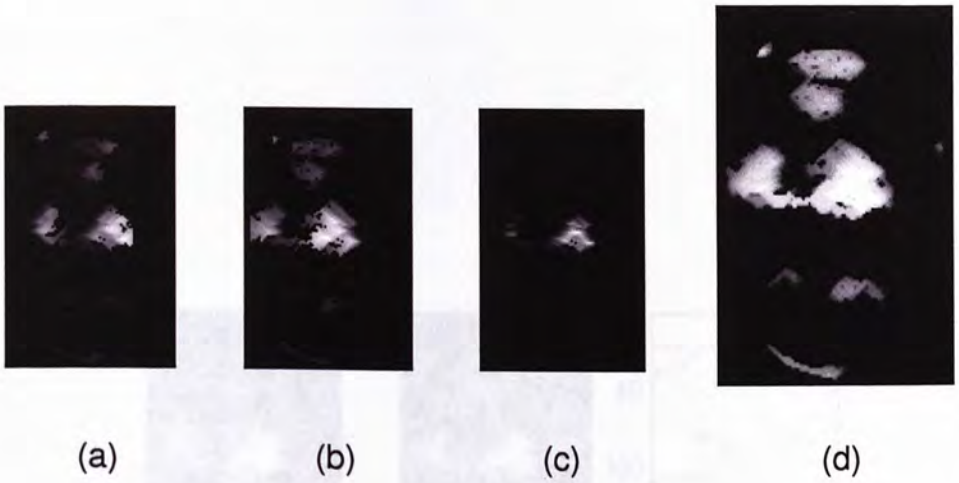


Figure 3.4: (a) L (b) R (c) ML (d) ML_{eq}

Figure 3.4 shows the left and right accumulated symmetry maps, the linked symmetry map, and the linked symmetry map after histogram equalization, of the symmetry map in Figure 3.3(b).

The width and horizontal location of the face box can be determined by projecting ML_{eq} onto the horizontal axis, as shown in Figure 3.5(b). We omit the projection values that are smaller than 0.1 of the maximum projection value to remove the noise, as shown in Figure 3.5(d). The remaining nonzero part in the projection is taken as the horizontal location of the box. The linked symmetry map after horizontal truncation is shown in Figure 3.5(c).

From the experiments, we found that the box height is slightly larger than (i.e., about $1.0 \sim 1.3$ of) the width. Therefore, from the horizontally truncated ML_{eq} (see Figure 3.5(c)), we locate the box vertically as the part whose height is 1.2 of the width and contains the maximum area of vertical projection. If the vertical

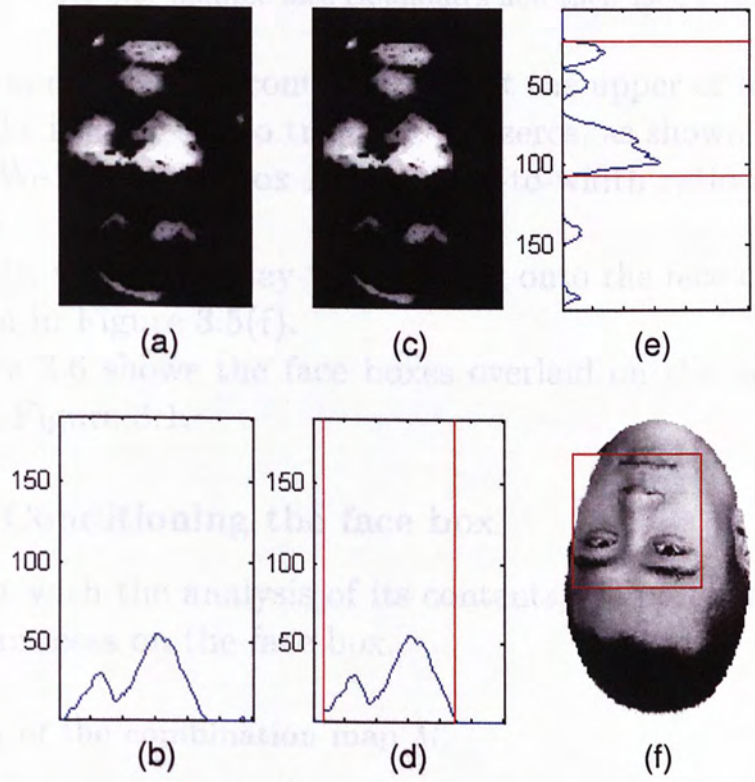


Figure 3.5: (a) ML_{eq}, (b) horizontal projection, (c) horizontally truncated ML_{eq}, (d) horizontal projection omitting small values, (e) vertical projection of (c), (f) face candidate with the face box located.

Figure 3.6: Sample face candidates and their face boxes

projection of this part contains zeros at the upper or lower side, the height is adjusted to truncate the zeros, as shown in Figure 3.5(e). We discard a box if its height-to-width ratio is smaller than 0.5.

Finally, we can overlay the face box onto the face candidate, as shown in Figure 3.5(f).

Figure 3.6 shows the face boxes overlaid on the face candidates in Figure 3.1.

3.1.2 Conditioning the face box

To assist with the analysis of its contents, we perform a conditioning process on the face box.

Creation of the combination map M_c

The eyes, the eyebrows and the mouth, have three common features:

1. They are darker than the surrounding face regions.
2. They include or are bounded by strong horizontal gradients.
3. They are isotropically symmetric.

Based on these three common features, we combine the following three measures on the facial feature box that give high responses accordingly:

1. The bright-dark exchanged intensities $max(I) - I$, where I is the intensity.
2. The horizontal gradients $\frac{\partial}{\partial x}\nabla I$.
3. The isotropic symmetry map M.

We next filter the above three measures by first introducing

$$m = max(3, [\frac{w}{10}]) \text{ and } n = max(3, [\frac{w}{14}])$$

where $[\cdot]$ means taking the nearest integer value, $max()$ means taking the larger value, and w is the box width. Hence both m and n are no smaller than 3. These two integers define the filter sizes.

- The intensities (see Figure 3.7(a)) are median filtered at every $m \times n$ pixels, so that some bright points inside the eyes or an opening mouth can be filled with surrounding black pixels, and some scattered dark pixels in the face region can also be removed. After median filtering, we apply lighting correction [14] and histogram equalization to improve the contrast. Finally, we interchange the brightness and the darkness. An example is shown in Figure 3.7(d).
- The horizontal gradients (see Figure 3.7(b)) are vertically averaged at every $1 \times 2n$ pixels. Since large horizontal gradients appear at the upper and lower bounds of the eyes, the eyebrows, and the mouth, the vertical averaging (or blurring) process can fill the points in these facial feature regions with high values. An example is shown in Figure 3.7(e)
- The isotropic symmetry map (see Figure 3.7(c)) is median filtered at every $2m \times 2n$ pixels so that the nearby high symmetry magnitudes in each cluster are connected and

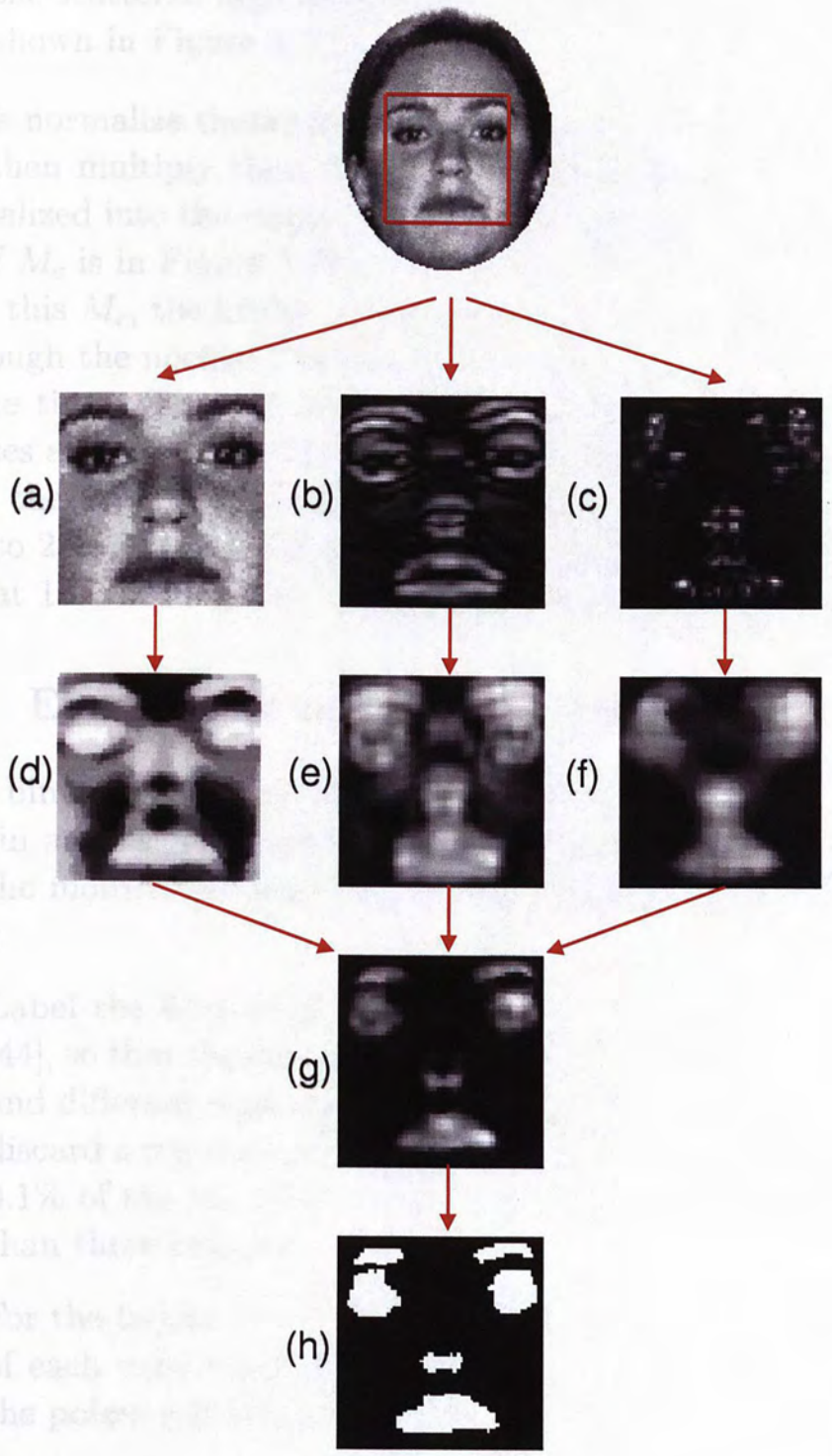


Figure 3.7: (a) Intensities (b) horizontal gradients (c) symmetry map (d) bright-dark interchanged intensities (e) blurred horizontal gradients (f) median-filtered symmetry map (g) combination map M_c (h) binarized box

the scattered high magnitudes are removed. An example is shown in Figure 3.7(f).

We normalize these three filtered images into the range $[0, 1]$ and then multiply them together. The resulting map is again normalized into the range $[0, 1]$. Let this map be M_c. An example of M_c is in Figure 3.7(g)

In this M_c, the bridge of the nose does not have high values. Although the nostrils can have high values, they do not appear all the time. Thus M_c is mainly designed to provide high responses at the regions of the eyes, the eyebrows and the mouth. From the experiments, we found that these regions cover from 15% to 20% of the total box area. Here, we binarize M_c at 0.18 so that 18% of its pixels are 1's, as shown in Figure 3.7(h).

3.2 Eye-mouth triangle search

If the binarized box (see Figure 3.7(h)) contains a face, it should contain at least one triangle formed by the eyes (or eyebrows) and the mouth. We search for such a triangle by the following steps:

1. Label the 4-connected regions of "1" in the binarized box [44], so that the points in each region have the same number and different regions are labelled by different numbers. We discard a region if its size (number of pixels) is smaller than 0.1% of the box size. Also a box is discarded if it has less than three regions.

2. For the largest six (at most) regions, calculate the position of each region center (x_i, y_i) as the average coordinates of the points inside the region.

3. Calculate the width of each region rw_i as the absolute difference of the leftmost and the rightmost x-coordinates of

the region.

4. Determine for every two regions, i and j, whether they belong to an eye (or eyebrow) pair. An eye pair must satisfy the following geometric constraints.

   - The line joining the two region centers is approximately horizontal, i.e.,
   $$\frac{|y_i - y_j|}{|x_i - x_j|} < 0.5$$
   - The two centers are symmetric with respect to the vertical midline of the box, i.e.,
   $$\frac{|\frac{x_i+x_j}{2} - \frac{w}{2}|}{w} < 0.25$$
   and
   $$(x_i - \frac{w}{2})(x_j - \frac{w}{2}) < 0$$
   where w is the box width.
   - The two regions should expand most of the box width, i.e.,
   $$\frac{|x_i - x_j| + \frac{rw_i}{2} + \frac{rw_j}{2}}{w} > 0.5$$
   - The widths of two regions should not differ much, i.e.,
   $$0.25 < \frac{rw_i}{rw_j} < 4$$

5. Discard the box if there is no eye (or eyebrow) pair found. Otherwise, search for the mouth, region k, for each eye pair, region i and j. Denote the Euclidean distance of two region centers, (x_p, y_p) and (x_q, y_q), as d_{pq}, and so on. Use the following constraints to search for the mouth given the eye pair (i, j).

   - The mouth region k cannot be i or j.

- Distance constraints:

$$\frac{d_{ki} - d_{kj}}{d_{ij}} < 0.6$$

$$-0.3 < \frac{d_{ki} - d_{ij}}{d_{ij}} < 1.3$$

and

$$-0.3 < \frac{d_{kj} - d_{ij}}{d_{ij}} < 1.3$$

- The mouth width cannot be larger than the expand of the eye pair

$$rw_k < |x_i - x_j| + \frac{rw_i}{2} + \frac{rw_j}{2}$$

- The eye-mouth triangle should be taller than half of the box height

$$|y_k - mean(y_i, y_j)| < \frac{h}{2}$$

where $mean()$ means taking the average value and h is the box height.

6. Position each triangle (i, j, k) so that the line joining the two eyes (i, j) is horizontal and the eyes are on the top, to resolve the up-down ambiguity. Select the positioned rectangles for matching against face models. Note that each eye (or eyebrow) pair can have more than one candidate mouth (see top image in Figure 3.8). Discard the box if there is no triangle found.

Figure 3.8 labels the five triangles contained in the box in 3.7(f), and shows the rectangles containing the positioned triangles that are selected to match with the face model.

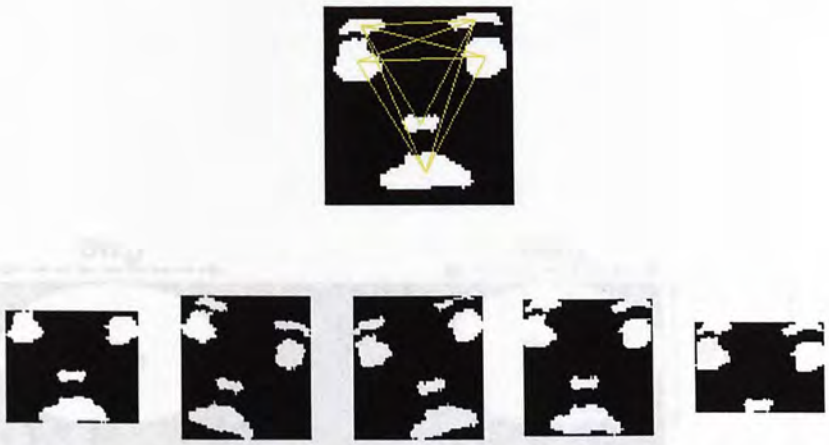


Figure 3.8: Labelled triangles and selected rectangles

3.3 Face model matching

A binary deformable face model is constructed for each selected rectangle (as shown in Figure 3.8) according to the positions and sizes of the candidate "eyes" and "mouth". Figure 3.9 shows the template used to construct the face model.

3.3.1 Face model construction

The sizes, ratios, and the geometric alignments of facial features in the template (see Figure 3.9) are defined by analyzing some face images and considering the template used in Maio and Maltoni's [40] system. Comparing to the template in [40], the eyes, eyebrows and mouth in our face models are wider because our models are to be matched with the combination map M_c while their template is to be matched with edge-like vectors in the directional image.

The face model is binary, where white points denote 1's and

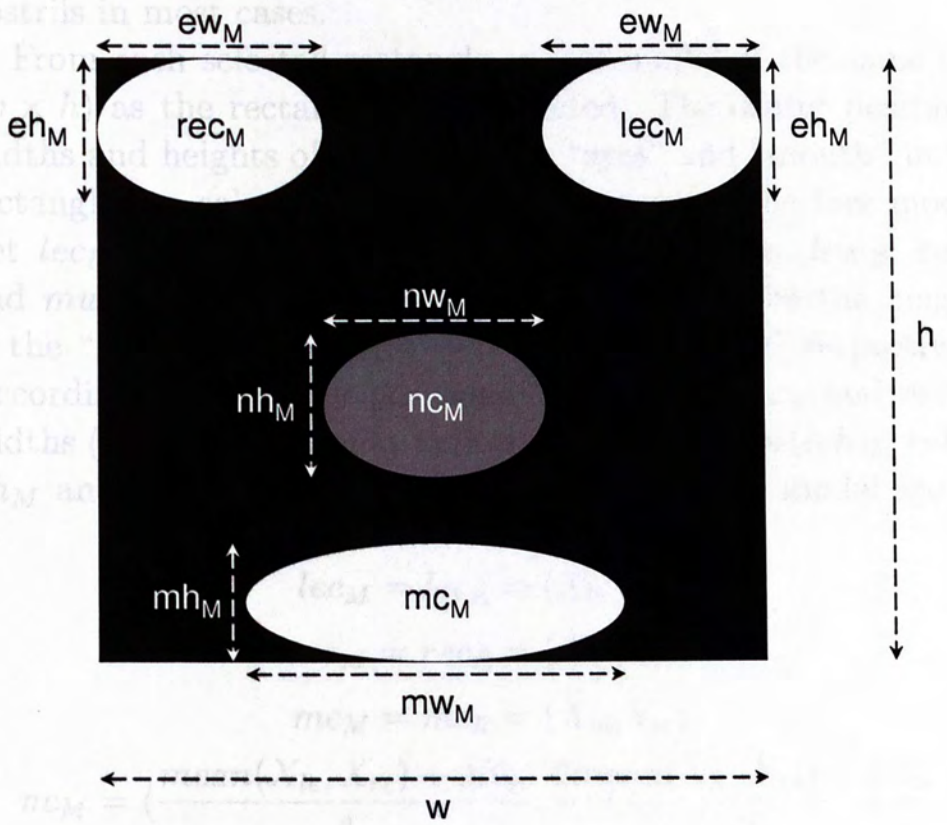


Figure 3.9: The template used to construct the face model

black points denote 0's. In Figure 3.9, the nose region is gray in color. This does not mean that its value is 0.5. We do not check the gray region, i.e., it is Boolean *don't care*. The reason is that the bridge of the nose does not appear in the selected rectangles and the nostrils appear only occasionally. The position of nostrils may vary according to different views of faces, hence we define a general region in the face model that can cover the nostrils in most cases.

From each selected rectangle, a face model of the same size ($w \times h$) as the rectangle is constructed. The center positions, widths and heights of the candidate "eyes" and "mouth" in the rectangle are calculated and used to construct the face model. Let lec_R, rec_R and mc_R be the center positions, lew_R, rew_R and mw_R be the widths, leh_R, reh_R and mh_R be the heights of the "left eye", the "right eye" and the "mouth" respectively. Accordingly, the center positions (lec_M, rec_M, nc_M and mc_M), widths (lew_M, rew_M, nw_M and mw_M) and heights (leh_M, reh_M, nh_M and mh_M) of the facial features in the face model are set as

$$lec_M = lec_R = (X_{le}, Y_{le})$$

$$rec_M = rec_R = (X_{re}, Y_{re})$$

$$mc_M = mc_R = (X_m, Y_m)$$

$$nc_M = (\frac{mean(X_{le}, X_{re}) + 3X_m}{4}, \frac{2mean(X_{le}, X_{re}) + 3X_m}{5})$$

$$ew_M = w/3$$

$$eh_M = mean(mean(leh_R, reh_R), h/5)$$

$$mw_M = mean(1.8ew_M, mw_R)$$

$$mh_M = mean(0.8eh_M, mh_R)$$

where w and h are the width and height of the rectangle and also the face model, $mean()$ means taking the average value.

3.3.2 Confidence of detection

We match each selected rectangle, R, with each of the four face models. A confidence value, C, is calculated as:

$$C = \frac{\sum R \bullet M}{\sum M} - \frac{\sum R \oplus M}{\sum \overline{M}}$$

where $\bullet$ = AND and $\oplus$ = EXCLUSIVE OR, $\overline{M}$ = NOT M, and M is the binary face model.

The first term in the confidence value measures the commonness while the second term measures the difference, so that the confidence value ranges from -1 to 1 theoretically. If the maximum confidence value of the selected rectangles in a face box is larger than a predetermined threshold (e.g. 0.5), a face is detected.

For the face box in Figure 3.8, the first selected rectangle has the maximum confidence, $C = 0.802 - 0.099 = 0.703$, so that the contained triangle is a detection. The vertices of the triangle are labelled on the original image as the eyes (the circles) and the mouth (the cross) accordingly in Figure 3.10.

3.4 Dealing with overlapped detections

In the first stage, the detector may locate multiple elliptical face candidates for the same face. Therefore, in the face verification stage, there could be overlapped detections. An example is shown in Figure 3.11, where each of the three elliptical candidates has a detection.

To eliminate overlapped results, we discard a detection i in an ellipse E_i if there exists another detection j such that $C_j > C_i$ and E_j overlaps more than half the area of E_i. Therefore, only the detection in Figure 3.11(a) will be included in the final detection result of the image.

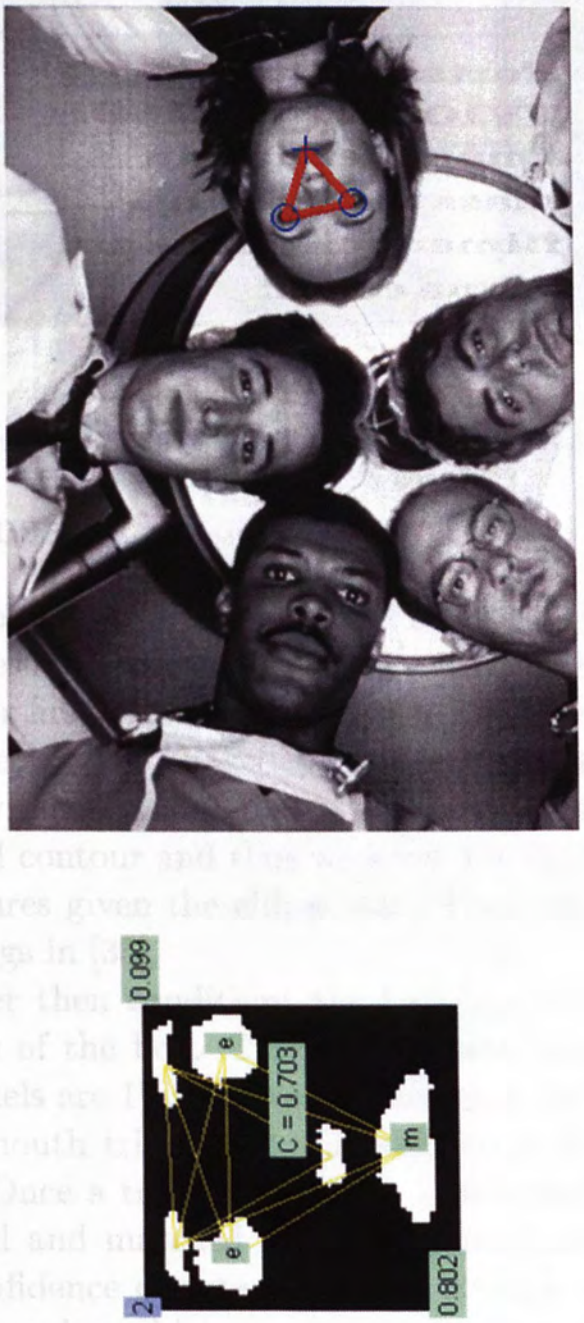


Figure 3.10: An example of detection

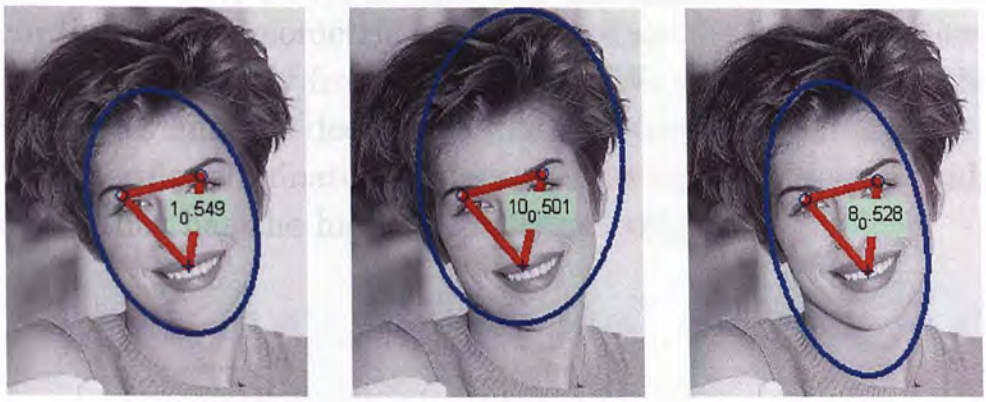


Figure 3.11: An example of overlapped detections

3.5 Discussion

In this chapter, we have introduced a face verifier to check the face candidates in the second stage of the proposed face detector.

The verifier first locates a face box in the face candidate ellipse by projecting the linked symmetry map measured by a generalized symmetry operator [35]. A face candidate ellipse is a face or head contour and thus we know the approximate sizes of facial features given the ellipse size. Therefore, it is easy to use the settings in [35].

The verifier then conditions the face box to create a combination map of the box. This map is next binarized so that 18% of its pixels are 1's. The verifier searches the binarized box for any eye-mouth triangle(s) that satisfies a set of geometric constraints. Once a triangle is found, a deformable face model is constructed and matched against the positioned triangle to check the confidence of detection. The triangle with the highest confidence value which exceeds a pre-determined threshold is taken as a face detection.

The geometric constraints for eye-mouth triangle search is

basic and not restrictive. We did not assign any cost value according to the geometric alignments since the faces and views of faces vary greatly from case to case. We simply select all the possibilities and the decision is made by face model matching.

The verifier eliminates overlapped detections by choosing only the one that has the highest confidence value.

□ **End of chapter.**

Chapter 4

Experiments

In this chapter, we access the performance of the proposed face detector through experimental results.

4.1 The test sets

The proposed face detector has been tested on 570 images from the MIT Database (see Table 1.1), 49 images from the CMU Rotated Test Set (a subset of the CMU Test Set in Table 1.1), and 16 images from my own collections. All faces in the test images are frontal or of small out-of-plane rotations.

In the MIT Database, each image contains only one face and all the faces are of 16 people. 9 images of each people are in the size of 128 $\times$ 120 pixels. 27 (or 26) images of each people are in the size of 256 $\times$ 240. The illumination conditions, scales and head orientations are different in the images of the same people, but the variation of the facial feature appearances is small because there are only 16 peoples in all 570 images. These 570 images with 570 faces from the MIT Database is the first test set.

The images from CMU and my own collections contain faces of arbitrary scales and in-plane rotations. Most contains more than one faces and the faces are of different people. These 49 +

16 = 65 images with 138 faces is the second test set.

Some sample test images and the detection results are shown in Figure 4.1 and Figure 4.2.

4.2 Experimental results

In Figure 4.1 and 4.2, the face candidate ellipses (CE's), and the eye-mouth triangles (T's) are all labelled on the faces by the detector. However, we can see that the positions of facial features are not always precisely located. Considering that facial feature localization is not the main purpose of our detector, we allow small deviations when collecting the detection results.

The miss of detection can be caused by the edge merging algorithm that fails in finding the CE's for some faces, and we call this kind of failure as CE_{miss}. There are also some cases in which the face verifier fails to find the corresponding T's, and we call this T_{miss}.

Table 4.1 shows the results obtained from the two test sets.

Table 4.1: Results on the two test sets.

Test Set	No. of faces	No. of CE's	No. of CE_{miss}'s	No. of T_{miss}'s
MIT	570	7694	28	18
CMU+own	138	1757	4	5

4.2.1 The ROC curves

To illustrate the performance of the detector in a more comparable way, Receiver Operating Characteristic (ROC) curves are drawn on the test sets by varying the threshold of confidence (i.e., confidence of detection described in Chapter 3.3.2).

For the first test set (MIT), the proposed detector can detect 335 faces out of 570 (58.77%) with 7 false alarms (or a false alarm rate of 7/7694 = 0.09%), for a threshold of 0.54. Changing

Figure 4.1: sample test images from MIT and the detection results

Figure 4.2: sample test images from CMU and the detection results

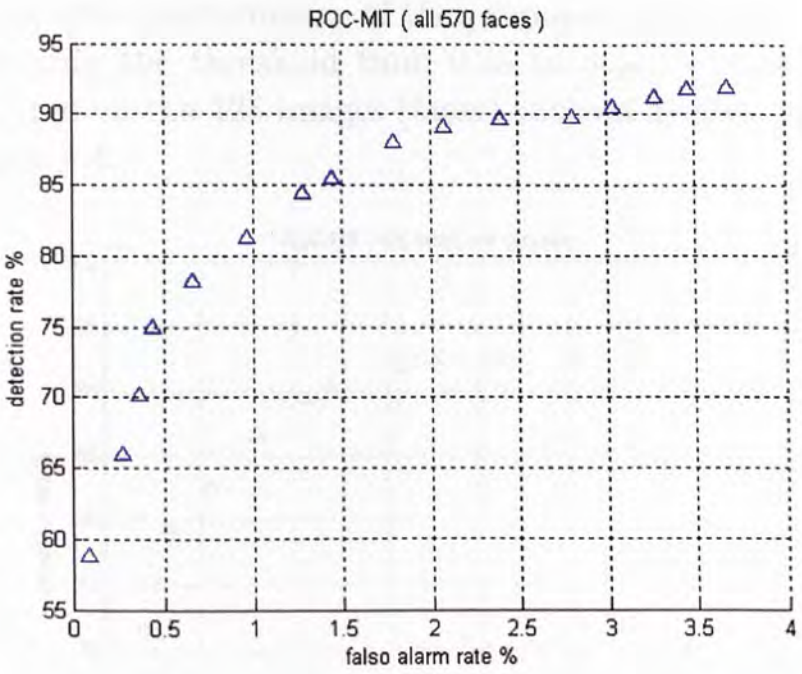


Figure 4.3: The ROC curve of all 570 images (with 570 faces) from the MIT Database

the threshold to 0.24, it can detect 524 faces (91.93%) while introducing 283 false alarms (or a false alarm rate of 283/7694 = 3.68%). Figure 4.3 shows the ROC curve of all 570 images from the MIT Database.

Recall that all 570 images from MIT are taken on 16 persons only. Among them, there are 4 persons wearing glasses. Under particular illumination conditions, the glasses are flaring lights which make the contents near the eyes chaotic. The (binarized) combination map (i.e., M_c described in Chapter 3.1.2) has problems when dealing with the "glasses-flaring" case, and more T_{miss}'s are caused as a result. Since all the images are taken under similar environment, the images for these 4 persons wearing glasses have this problem. As these images cover $142/570 \approx 25\%$ of all 570 images, we divide the first test set into two cases, i.e., "with glasses" and "without glasses", to further

investigate the performance of the proposed detector.

Changing the threshold from 0.24 to 0.54, we can draw a ROC curve on the 428 images (faces) without glasses, as shown in Figure 4.4.

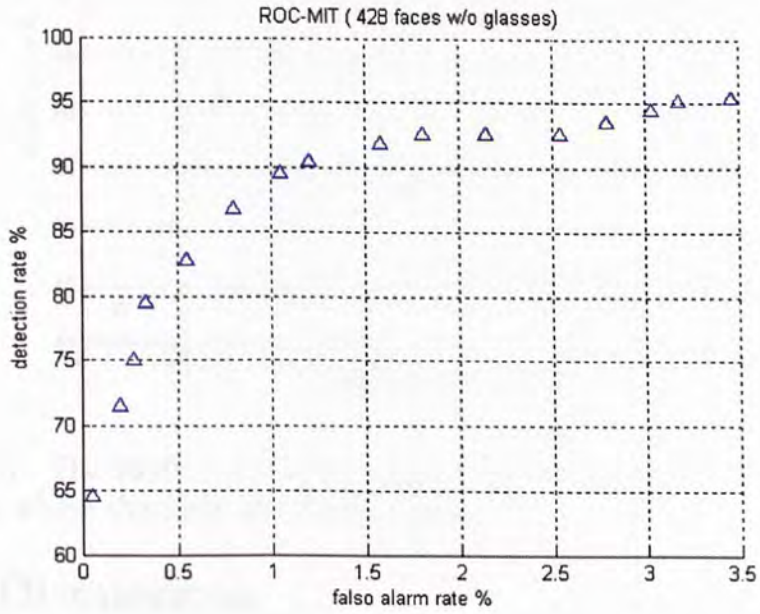


Figure 4.4: The ROC curve of 428 images (with 428 faces) from the MIT Database where the faces are without glasses

Changing the threshold from 0.32 to 0.51, we can draw a ROC curve on the 142 images (faces) with glasses, as shown in Figure 4.5.

From Figure 4.4 and 4.5, we can see that the performance of the detector on the faces without glasses is much better than those with glasses, as expected.

For the second test set (CMU+own), the proposed detector can detect 115 faces out of 138 without false alarms, which yields a detection rate of 83.33%, for a threshold of 0.43. Changing the threshold to 0.32, it can detect 129 faces (93.48%) while introducing 28 false alarms (or a false alarm rate of $28/1757 = 1.59\%$). Figure 4.6 shows the ROC curve.

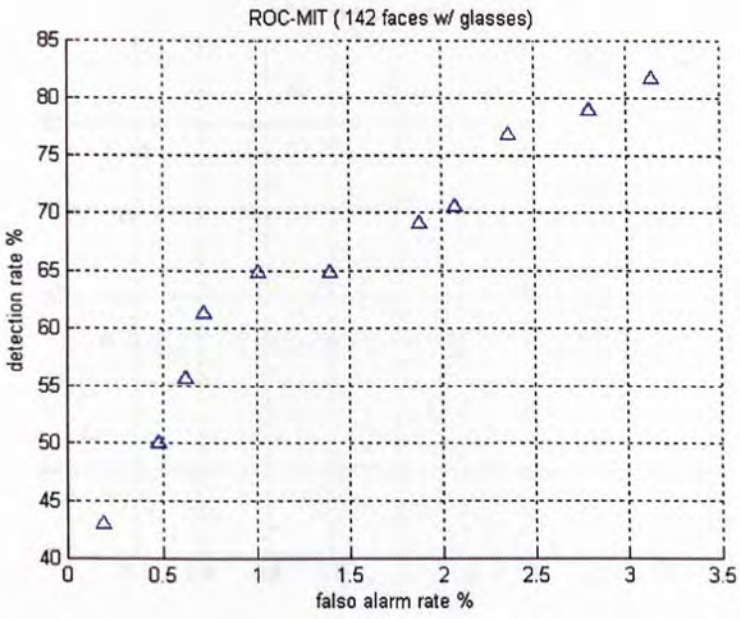


Figure 4.5: The ROC curve of 142 images (with 142 faces) from the MIT Database where the faces are wearing glasses

4.3 Discussions

Comparing Figure 4.3 (or 4.4) with Figure 4.6, we can see that the detector performs better in the second test set than in the first one. The main reason is that the illumination conditions of the CMU images and my own collections are better than most of the MIT images. In other words, the face verifier still needs to be improved to deal with poor illumination conditions.

In the second test set (CMU+own), the edge merging algorithm generates 1757 elliptical face candidates that cover 134 of 138 faces, i.e., there are 4 CE_{miss}'s. There are other 5 T_{miss}'s because the face verifier fails to find the corresponding eye-mouth triangles. This is the reason why the detector can only achieve a maximum of $138-4-5=129$ detections, or a highest detection rate of 93.48% (see Figure 4.6).

From Figure 4.7 to Figure 4.11, we show the test images that

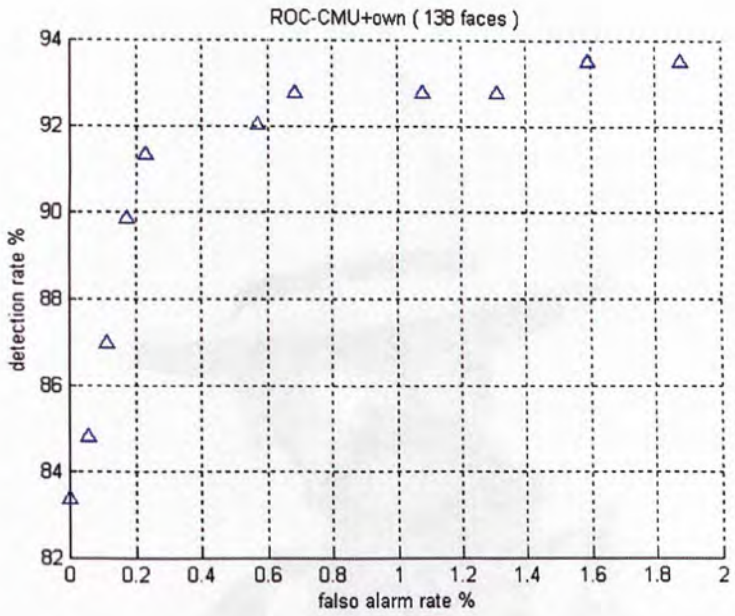


Figure 4.6: The ROC curve of 65 images (with 138 faces) from CMU and own collections

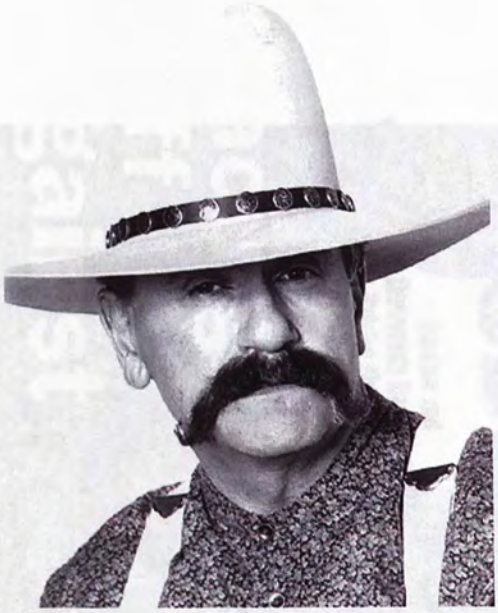
contain these CE_{miss}'s and T_{miss}'s.

From these images, we can see that some of the reasons causing CE_{miss}'s and T_{miss}'s are:

- poor illumination condition
- very small face size and low image resolution
- the face is occluded, or wears strange headgear or beards.

To reduce the CE_{miss}'s, we can decrease the double thresholds of the Canny edge detector to produce more edges, or apply the curve merging process on more straight curves and keep very short curves for merging in the edge merging algorithm. But by doing so, we will also introduce more false face candidates, making the face verification more difficult and the whole process more time-consuming.

Some of the T_{miss}'s are actually caused by the inaccurate positions and/or poor orientations of their face candidate ellipses.



(a)

(b)

Figure 4.7: (a) A test image that causes one CE_{miss}, image size: 312×387 (b) A test image that causes one CE_{miss} (the largest face in image - other small faces are out of our scope), image size: 375×375

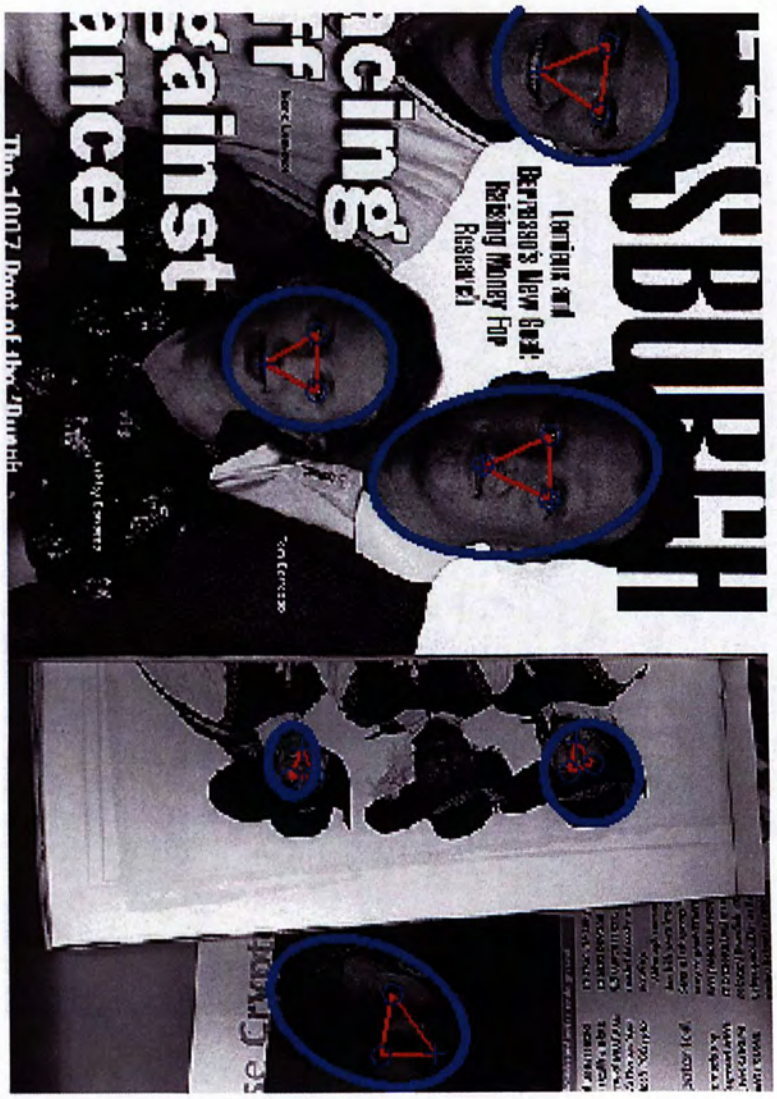


Figure 4.8: A test image that contains one CE_{miss} (the face without ellipse), image size: 410×580

Figure 4.9: A test image that contains one CE_{miss} (the face without ellipse) and one T_{miss} (the face with an ellipse but without a triangle), image size: 499×402, rotated 90^0 to fit the page

Figure 4.10: A test image that contains one T_{miss} (the face with an ellipse but without a triangle), image size: 640×438, rotated 90^0 to fit the page

(a)

(b)

(c)

Figure 4.11: (a) A test image that contains one T_{miss} (the face with an ellipse but without a triangle - the piggy is out of our scope), image size: 375×531 (b) A test image that causes one T_{miss}, image size: 180×166 (c) A test image that causes one T_{miss} (the detector labels the nose as an eye, the left eye as the mouth), image size: 252×426

Thus the triangles of the true face either cannot pass the geometric constraints, or the detected triangle has the eyes and mouth wrongly labelled (see Figure 4.11(c)).

Comparing to the learning-based face detectors, our detector does not require training process. We cannot claim that our detector achieves a better performance than the ROC curves in Figure 1.5, since our detector is not tested on the same test sets. However, our test set does cover a wide variety of image and face conditions. If our detector is to be integrated into a particular application, all the thresholds can be trained according to the specific image (or frame) conditions to achieve a better performance.

Comparing to other feature-based face detectors (e.g. Maio and Maltoni's [40]), our detector can detect multiple frontal faces in arbitrary sizes and in-plane rotations hence is much more versatile. Since both the edge merging and the face verification process can be done in parallel, and due to simplicity at the algorithm level, our detector is also efficient.

□ **End of chapter.**

Chapter 5

Conclusions

5.1 Conclusions

This thesis has presented a two-stage feature-based approach to detect in-plane rotated faces on grayscale images with complex backgrounds. In the first stage, edges from the Canny edge detector are conditioned and then merged. Ellipses are next fitted to the merged curves. Those whose fitting errors are below a predetermined threshold are then passed to the face verification stage. This stage matches a binary face model with each eye-mouth triangle found in the face candidates for detection.

This method avoids the need for exhaustive searches in the image pyramid. The algorithms are simple and the computational requirement is low. Hence the detector has a high potential for real-time applications. For a test set of 65 images containing 138 near-frontal faces in arbitrary scales and in-plane rotations, this detector, depending on the threshold chosen, can give a detection rate of 83.33% with no false alarms, or a detection rate of 93.48% with 28 false alarms.

However, it is not easy to compare a face detector with another. The main difficulties come from the following aspects.

- In order to fairly evaluate face detection approaches, it is important to use a standard and representative test set for

experiments. Although many face detection methods have been developed, only a few of them have been tested on the same data set.

- The common test sets, e.g., the test sets from MIT and CMU, provide some basis for comparison, but since there are no agreements on the number of faces in the test sets, especially the CMU test set, it is hard to compare the detection rates obtained by different detectors.
- The detection results are collected (manually or automatically) under different considerations, making the comparison unfair.
- There are no standard evaluation procedures. The numbers of scanning windows vary in different learning-based approaches because they are designed to detect faces within different size ranges, so that the false alarm rates are not comparable.
- Since the reported results are always based on different tuning parameters, the results are not comparable unless ROC curves on the same test sets are reported.

The human face is a dynamic object that changes over time and varies in pose. It is still a hard problem to develop a robust face detection algorithm. This thesis provides an alternative way, rather than the traditional exhaustive search in the image pyramid, to limit the search region within face candidate ellipses for efficient detection of multiple rotated frontal faces.

5.2 Suggestions for future work

The proposed detector has a flexible structure so that it can be enhanced at any stage to improve the performance. For example,

an adaptive corner detection method may help the edge merging algorithm to find the face candidate ellipses for very small faces when the quantization noise is large. More face models could be constructed to cover faces in different conditions, and they are not necessarily binary if a better representation than the binarized combination map (described in Chapter 3.1.2) can be developed. Other feature-based approaches (that were designed to detect a single face) can also be incorporated into the face verifier.

Using the same framework, by modifying the cost function for curve merging and with a more flexible face verifier (e.g. using more face models), our detector can also be upgraded to detect the faces in out-of-plane rotations.

Although the proposed detector is designed for grayscale images, it can be easily integrated with other approaches when more information, such as color and motion, is available. In that case, the performance, in terms of both accuracy and speed, will certainly improve.

□ **End of chapter.**

List of Original Contributions

- A curve merging process including a cost function for merging in the defined search region.
- Application of fitting merged face contours with ellipses to a face detection approach.
- Creation and conditioning of the face box.
- Extensible binary face models for efficient face box matching.

Bibliography

[1] Y. Yacoob and L. Davis, "Computing spatio-temporal representations of human faces," in *Proc. 1994 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, pp. 70–75, 1994.

[2] C. H. Lee, J. S. Kim, and K. H. Park, "Automatic face location in a complex background using motion and color information," *Pattern Recognition*, vol. 29, pp. 1877–1889, 1996.

[3] B. Leroy, I. L. Herlin, and L. D. Cohen, "Face identification by deformation measure," in *Proc. 13th Int'l Conf. on Pattern Recognition*, pp. 633–637, 1996.

[4] J. Cai, A. Goshtasby, and C. Yu, "Detecting human faces in color images," in *Proc. 1998 Int'l Workshop Multi-Media Database Management Systems*, pp. 124–131, 1998.

[5] M. H. Yang and N. Ahuja, "Gaussian mixture model for human skin color and its application in image and video databases," in *Proc. SPIE: Storage and Retrieval for Image and Video Databases VII*, vol. 3656, pp. 458–466, 1999.

[6] S. L. Phung, A. Bouzerdoum, and D. Chai, "A novel skin color model in ycbcr color space and its application to human face detection," in *Proc. ICIP Int'l Conf. on Image Processing*, pp. 289–291, 2002.

[7] H. I. Kim, S. H. Lee, and N. I. Cho, "Rotation-invariant face detection using angular projections," *Electronics Letters*, vol. 40, no. 12, 2004.

[8] H. Wu, Q. Chen, and M. Yachida, "Face detection from color images using a fuzzy pattern matching method," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 21, no. 6, pp. 557–563, 1999.

[9] R.-L. Hsu, M. Abdel-Mottaleb, and A. K. Jain, "Face detection in color images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 696–706, 2002.

[10] M. Kapfer and J. Benois-Pineau, "Detection of human faces in color image sequences with arbitrary motions for very low bit-rate videophone coding," *Pattern Recognition Letters*, vol. 18, pp. 1503–1518, 1997.

[11] T. Sakai, M. Nagao, and S. Fujibayashi, "Line extraction and pattern detection in a photograph," *Pattern Recognition*, vol. 1, pp. 233–248, 1969.

[12] T. Sakai, M. Nagao, and T. Kanade, "Computer analysis and classification of photographs of human faces," in *Proc. First USA-Japan Computer Conference*, p. 2.7, 1972.

[13] R. Chellappa, C. L. Wilson, and S. Sirohey, "Human and machine recognition of faces: a survey," *Proc. IEEE*, vol. 83, no. 5, pp. 705–740, 1995.

[14] E. Hjelmas and B. K. Low, "Face detection: a survey," *Computer Vision and Image Understanding*, vol. 83, no. 3, pp. 236–274, 2001.

[15] M. H. Yang, D. J. Kriegman, and N. Ahuja, "Detecting faces in images: a survey," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 34–58, 2002.

[16] K. K. Sung and T. Poggio, "Example-based learning for view-based human face detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 39–51, 1998.

[17] M. H. Yang, N. Ahuja, and D. Kriegman, "Face detection using mixtures of linear subspaces," in *Proc. Fourth Int'l Conf. Automatic Face and Gesture Recognition*, pp. 70–76, 2000.

[18] M. H. Yang, D. Roth, and N. Ahuja, "A SNoW-based face detector," in *Advances in Neural Information Processing Systems 12* (S. A. Solla, T. K. Leen, and K. R. Müller, eds.), pp. 855–861, MIT Press, 2000.

[19] H. Schneiderman and T. Kanade, "Probabilistic modeling of local appearance and spatial relationships for object recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 45–51, 1998.

[20] H. Schneiderman and T. Kanade, "A statistical method for 3d object detection applied to faces and cars," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, pp. 746–751, 2000.

[21] H. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23–38, 1998.

[22] H. Rowley, S. Baluja, and T. Kanade, "Rotation invariant neural network-based face detection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 38–44, 1998.

[23] A. Z. Kouzani, "Locating human faces within images," *Computer Vision and Image Understanding*, pp. 247–279, 2003.

[24] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: an application to face detection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 130–136, 1997.

[25] R. Xiao, M. J. Li, and H. J. Zhang, "Robust multipose face detection in images," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 14, no. 1, pp. 31–41, 2004.

[26] P. Viola and M. J. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, pp. 511–518, 2001.

[27] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[28] S. Z. Li, L. Zhu, Z. Q. Zhang, and et al, "Statistical learning of multi-view face detection," in *Proc. 7th European Conf. Computer Vision*, LNCS 2353, pp. 67–81, 2002.

[29] B. Wu, H. Ai, C. Huang, and S. Lao, "Fast rotation invariant multi-view face detection based on real Adaboost," in *Proc. Sixth Int'l Conf. Automatic Face and Gesture Recognition*, pp. 79–84, 2004.

[30] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Computational Learning Theory: Eurocolt 95*, pp. 23–37, Springer-Verlag, 1995.

[31] B. H. Jeon, S. U. Lee, and K. M. Lee, "Face detection using the 1st-order RCE classifier," in *Proc. IEEE Conf. Image Processing*, vol. 2, pp. II–125–II–128, 2002.

[32] J. Wang and T. Tan, "A new face detection method based on shape information," *Pattern Recognition Letters*, vol. 21, pp. 463–471, 2000.

[33] G. Yang and T. S. Huang, "Human face detection in complex background," *Pattern Recognition*, vol. 27, no. 1, pp. 53–63, 1994.

[34] X. G. Lv, J. Zhou, and C. S. Zhang, "A novel algorithm for rotated human face detection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, pp. 760–765, 2000.

[35] D. Reisfeld and Y. Yeshurun, "Preprocessing of face images: detection of features and pose normalization," *Computer Vision and Image Understanding*, vol. 71, no. 3, pp. 413–430, 1998.

[36] S. R. Gunn and M. S. Nixon, "A dual active contour for head and boundary extraction," in *IEE Colloquium on Image Processing for Biometric Measurement*, p. 6/1, 1994.

[37] A. Yuille, P. Hallinan, and D. Cohen, "Feature extraction from faces using deformable templates," *International Journal of Computer Vision*, vol. 8, no. 2, pp. 99–111, 1992.

[38] L. C. De Silva, K. Aizawa, and M. Hatori, "Detection and tracking of facial features by using a facial feature model and deformable circular template," *IEICE Trans. Information and Systems*, vol. E78-D, no. 9, pp. 1195–1207, 1995.

[39] K. C. Yow and R. Cipolla, "Feature-based human face detection," *Image and Vision Computing*, vol. 15, no. 9, pp. 713–735, 1997.

[40] D. Maio and D. Maltoni, "Real-time face location on gray-scale static images," *Pattern Recognition*, vol. 33, pp. 1525–1539, 2000.

[41] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[42] H. Freeman, "Computer processing of line-drawing images," *Computing Surveys*, vol. 6, no. 1, pp. 57–97, 1974.

[43] W. Gander, G. H. Golub, and R. Strebel, "Fiting of circles and ellipses: least squares solution," *BIT 34*, pp. 556–577, 1994.

[44] R. M. Haralick and G. S. Linda, *Computer and Robot Vision*, vol. I, pp. 28–48. Addison-Wesley, 1992.